1998-03-01

Model analysis of energy spreading loss off the Carolina Coast for tactical active sonars

Smith, Peter E.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

MODEL ANALYSIS OF ENERGY SPREADING LOSS OFF THE CAROLINA COAST FOR TACTICAL ACTIVE SONARS

by

Peter E. Smith

March, 1998

Thesis Advisor: Robert H. Bourke
James H. Wilson

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1998	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE MODEL ANALYSIS OF ENERGY SPREADING LOSS OFF THE CAROLINA COAST FOR TACTICAL ACTIVE SONARS

5. FUNDING NUMBERS

6. AUTHOR(S)
Smith, Peter E. in conjunction with Bourke, R.H. and Wilson, J. H

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Energy spreading loss (ESL) is the reduction of the transmitted pulse energy level by spreading of the pulse in time due to multipath propagation. This energy spreading will reduce the effectiveness of mid-frequency tactical sonars. The U.S. Navy training areas of Long Bay and Onslow Bay off the Carolina Coast were chosen for the study of ESL to provide contrasts in many of the geoacoustic properties that can change ESL. Inputs were varied by source depth, receiver depth, sound speed profile(SSP), bathymetry, and geoacoustic properties. The computer model FEPE_SYN calculated the ocean transfer function (OTF) for the modeled environment in the frequency domain. The time domain output pulse was calculated using the OTF, an input pulse, and an inverse discrete Fourier transform. Using the same energy as the output pulse, a compressed pulse was created with the same shape as the input pulse. ESL was determined by comparing the peak level of the output pulse to the peak level of the compressed pulse. A mismatch loss (MML) was calculated by comparing the maximum values from the correlation of the input pulse with the output pulse and compressed pulse.

The ESL of the output pulse was dependent on several factors. Absorptive (silt/clay) sediment sea beds had average ESL values 3 dB less than that of compacted sand. The compacted sand bottom was also compared to an even more reflective sediment, a limestone sediment layer. ESL values were higher by an additional 3 dB for the limestone bottom. Minimum ESL levels were found when the source and target were at the same depth. Changing source and target depths (e.g., cross layer) could increase ESL levels up to 8 dB from the minimum ESL level. The impact of using a range-dependent SSP vice constant SSP was inconclusive in that ESL values could be larger or smaller by 3 dB compared to range-independent runs. Similar inconclusive results were obtained when actual bottom depths were employed vice a flat-bottom run. As found by Tanaka (1996), ESL was observed to rapidly increase in the first 1000 m and thereafter fluctuate around a mean value. This initial critical range is evidently site dependent but appears to be confined between 300 to 1000 m range.

14. SUBJECT TERMS Acoustics, Energy Spreading Los, ESL, Underwater System, FEPE, FEPE_SYN, Active Sonar, Transmission Loss, Mismatch Loss, MML, Time Domain Analysis

15. NUMBER OF PAGES 192

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

MODEL ANALYSIS OF ENERGY SPREADING LOSS OFF THE CAROLINA COAST FOR TACTICAL ACTIVE SONARS

Peter E. Smith
Lieutenant, United States Navy
B.S., University of Illinois, 1992

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN PHYSICAL OCEANOGRAPHY

from the

NAVAL POSTGRADUATE SCHOOL
March 1998

ABSTRACT

Energy spreading loss (ESL) is the reduction of the transmitted pulse energy level by spreading of the pulse in time due to multipath propagation. This energy spreading will reduce the effectiveness of mid-frequency tactical sonars. The U.S. Navy training areas of Long Bay and Onslow Bay off the Carolina Coast were chosen for the study of ESL to provide contrasts in many of the geoacoustic properties that can change ESL. Inputs were varied by source depth, receiver depth, sound speed profile(SSP), bathymetry, and geoacoustic properties. The computer model FEPE_SYN calculated the ocean transfer function (OTF) for the modeled environment in the frequency domain. The time domain output pulse was calculated using the OTF, an input pulse, and an inverse discrete Fourier transform. Using the same energy as the output pulse, a compressed pulse was created with the same shape as the input pulse. ESL was determined by comparing the peak level of the output pulse to the peak level of the compressed pulse. A mismatch loss (MML) was calculated by comparing the maximum values from the correlation of the input pulse with the output pulse and compressed pulse.

The ESL of the output pulse was dependent on several factors. Absorptive (silt/clay) sediment sea beds had average ESL values 3 dB less than that of compacted sand. The compacted sand bottom was also compared to an even more reflective sediment, a limestone sediment layer. ESL values were higher by an additional 3 dB for the limestone bottom. Minimum ESL levels were found when the source and target were at the same depth. Changing source and target depths (e.g., cross layer) could increase ESL levels up to 8 dB from the minimum ESL level. The impact of using a range-dependent SSP vice constant SSP was inconclusive in that ESL values could be larger or smaller by 3 dB compared to range-independent runs. Similar inconclusive results were obtained when actual bottom depths were employed vice a flat-bottom run. As found by Tanaka (1996), ESL was observed to rapidly increase in the first 1000 m and thereafter fluctuate around a mean value. This initial critical range is evidently site dependent but appears to be confined between 300 to 1000 m range.

TABLE OF CONTENTS

I. INTRODUCTION

Underwater acoustics is sometimes thought of as a relatively new science, but it does have a long history. Ancient Chinese fishermen would follow schools of fish by placing a bamboo stick in the water. Ceylon fishermen would signal each other several miles away by striking submerged jars (Hackmann, 1984). These are some earlier uses of sound in the water. Underwater acoustics remained in this operationally primitive state until the last century when many advances were made. In 1889, a bell and hydrophone system was developed as an aid to navigation. The underwater bells were placed near shoal water, lighthouses, or other aids to navigation where merchant ships would listen for the sound with early hydrophones (Hackmann, 1984). World War I saw the passive use of hydrophones in the search for submarines. Later in the war directional hydrophones were developed and installed. The British installed an active sensor almost immediately after the end of the war in the form of a single quartz transducer which created a searchlight pattern (Hackmann, 1984). This remained the basic sonar design through out World War II. The standard sonar installed on United States ships required the operator to search by turning a handwheel. If a return was heard, the range was noted by the flash of a rotating light (Urick, 1983). With the advances made throughout the war time period, British and American scientists were able to design sonars with multiple transducers aligned as vertical staves in 360 degree arrays. This enabled the sonar to conduct omnidirectional searches with beamforming by using delay circuits. This is the basic design behind the current sonar systems in use today (Hackmann, 1984).

Current advances in underwater sound technology are generally related to processing of the signal received from the transducer. With the use of computers, a steady transition from analog processing to digital processing has developed (Knight et al., 1981). Sonar systems can now exploit advances in other disciplines. Being similar to radar in principle, sonar can take advantage of the algorithmic developments, waveform processing, digital filtering, and Fourier transforms that have been developed for radar

applications (Knight et al., 1981). The high speed and large data storage capability of modern computing now permits research to be conducted in areas that were not even considered a few years ago.

A large portion of the current research has been focused on improving sonar performance in shallow water. This emphasis comes from the U. S. Navy's interest in littoral regions. As noted in the paper "...From The Sea", an enemy shallow water submarine can be of particular concern for naval forces operating in shallow water (O'Keefe et al., 1992). Sonar performance is highly variable and difficult to predict in shallow water "because many environmental acoustic parameters, such as the temperature and salinity of the water column, bathymetry, type of sediment, background noise, marine life, etc..., are highly variable, both temporally and spatially (Tanaka, 1996)." The major factor leading to the variability and uncertainty in sonar performance in shallow water is the multipath interaction caused by interaction of the acoustic signal with the bottom. To model the bottom interaction accurately requires that a full wave solution be used to estimate the sound propagation rather than the simpler ray acoustic theory used successfully for many decades in deep water environments. This study will focus on a specific factor that inherently reduces the effectiveness of tactical sonars in shallow water, termed energy spreading loss.

II. ENERGY SPREADING LOSS

The AN-SQS 26/53 sonar system has been the mainstay of the United States Navy surface antisubmarine (ASW) force for 35 years. The 26 system was first installed on the Bronstein class destroyers in 1963 (Bonds, 1987). The 53 system is a solid state version of the 26, and the C variant of the 53 is a digital version (Bonds, 1987). The knowledge that the performance of tactical sonar systems degrade in shallow water due to energy spreading loss (ESL) reaches back to the early 1960's. Stewart and Brandon (1967) introduced the term 'energy splitting loss' to describe the observable result of numerous arrivals being processed by the correlator. The energy split was attributed to the multipath effects of the medium. Wittenborn (1965) also noted that the splitting can occur in the frequencies being transmitted. The term 'splitting' was modified to 'spreading' by Weston (1965). He noted that the loss in peak level when a short ping was transmitted was similar to the correlation loss of a longer pulse. Weston also developed a prediction model for ESL based on a Gaussian distribution. This development was similar to the communication theory of the time (Kennedy, 1969). This early work on time spreading was discontinued for several of reasons. First, the funding for fundamental research and development ended. Another factor was that the SQS-26 system used a scan converter based upon an integration algorithm which reduced some losses caused by time spreading. ESL reemerged as topic of interest because of the 53C's use of peak picking logic (Bell, 1989).

Recently, the air-borne low frequency (ALFS) helicopter-deployed sonar system has emerged to compliment the 53 C in both deep and shallow water. The major disadvantage of the 53 C is its fixed source depth, usually within the mixed layer (ML), and ALFS has the capability of operating well below the ML depth. In this study the source and target depths are realistically varied above and below the ML to determine their impact on ESL. These analyzes apply when the ALFS sonar is present, preferably operating in concert with a 53C, to ensonify the shallow water column both above and below the ML.

Recent work on ESL focuses on the ability to predict the signal loss due to time spreading. Two researchers, Bell (1989) and Jones (1989), have modified Weston's Gaussian model. Bell assumed a Rayleigh distribution for calculations to model the energy spreading since current systems now use peak processors. Rather than using energy summations, these processors look for peaks in the correlation of the returned echo with the transmitted pulse. The Rayleigh distribution models the fluctuations that occur in a propagating signal. As seen in Figure 1, peaks can be exploited by peak processors that are not taken into account when a Gaussian distribution is used. The Gaussian distribution underestimates the processing capability of the peak processors. To obtain a statistically reliable result, Bell conducted a 100 run Monte Carlo simulation. He found that the ESL values dropped from a Gaussian prediction of 11 dB to an average of 7.3 dB for the Monte Carlo simulation (Bell, 1989). Jones added to this work by running the simulation over a variety of time-spread sigma (standard deviation) to time resolution ratios. Larger time spread sigmas required a larger resolution window to maintain the same ESL in the simulation. This resulted in an empirical equation for ESL:

$$\text{(Time Spread Sigma/ Time Resolution)} = \mu \qquad (1)$$

$$\mu < .2 \qquad \text{ESL} = 0$$

$$2 \leq \mu \geq 3 \qquad \text{ESL} = 4.68\,[\log(\mu)+.699)^{1.227}$$

$$\mu \geq 3 \qquad \text{ESL} = 2.47 + 6.8\log(\mu)$$

Another experiment measured ESL for the 53C sonar. From this experiment, Chan (1992) developed a model that could predict ESL which considered all the frequencies in the bandwidth being transmitted. The model was not considered successful in shallow water due to contamination by the noise field.

A third method was proposed to measure the mismatch loss (MML). This method compares the shape of the received and transmitted pulses rather than their energy levels. The equation:

$$\rho(t) = \frac{\int V_r(t) * V_p(\tau-\tau)\, dt}{[I\, V_r^2(t)\, dt\,]^{1/2} * [I\, V_p^2(t)\, dt\,]^{1/2}} \qquad (2)$$

produces a normalized coherence prediction where $V_r(t)$ is the replica pulse and $V_p(t)$ is the propagated or received pulse. Jensen and Sabbadini(1987) use a bottom interaction simulator and recorded MML levels for a low frequency active sonar. They used bottom impulse response functions provided by SACLANTCEN data, a 1/100 scale cylinder-shaped target with rounded end caps, a bistatic geometry, and two bottom bounces for a pulse emitted at various launch angles. Their results showed that MML can reach 5.3 dB for a wide band (0.5 - 5 kHz) 1.5 second LFM pulse using match filter processing.

Tanaka (1996) proposed a different method to quantify ESL. Since ESL is caused by the stretching of the transmitted pulse by definition, a pulse that is not deformed by time spreading will have no ESL. The reduction of the peak level of the received pulse due to time spreading was proposed as the new definition of ESL. This is graphically shown in Figure 2. The total loss, without signal processing to recover ESL, is measured from the peak of the transmitted pulse to the peak of the spread pulse. A hypothetical compressed pulse is defined as a theoretical value determined by combining all the energy of the spread pulse into one pulse having a similar shape to the transmitted pulse. Differences between the peak of the transmitted pulse and the compressed pulse are caused by the transmission losses of the ocean propagation medium. The difference between the transmission loss and total loss is attributed to ESL. This can also be viewed as the change in peak energy level of the compressed pulse compared to the peak energy of the spread pulse. A decibel measurement is made by taking the logarithm of the change in peak levels (Tanaka, 1996).

$$10*\log_{10}(E\text{compressed}_{max}/E\text{spread}_{max})\ [\text{dB}] \qquad (3)$$

Tanaka used this definition to evaluate the finite element parabolic equation (FEPE) model output for ESL. For the same tactical sonar frequencies evaluated in this paper, he evaluated the effects of a highly controlled environment on ESL. The FEPE model inputs were kept range independent. With a constant 64 m depth, he varied the sound speed profile for the water column and the geoacoustic values of the ocean floor. In range, ESL was found to increase significantly in the first 1600 m and then increase much more slowly beyond that distance. At any given range, Tanaka examined the depth dependence of ESL. He discovered fluctuations to be present in ESL but concluded that ESL was only slightly dependent on the depth of the target given a constant source depth. The dependence of ESL relied upon the location of the source relative to the target with a reduction in ESL noted if the source was at a similar depth to the target. The geoacoustics of the ocean bottom greatly affected ESL levels. Using bottom values used for Area Foxtrot, a shallow water test area off Long Island, Tanaka found that highly reflective bottoms had larger ESL levels. The reflective bottoms allowed more of the higher order modes to propagate. Because these modes each travel at their own group speed, the more modes that are present leads to increased time spreading of the transmitted pulse.

The total active sonar equation must be considered to assess sonar performance. It is important to note that ESL does not imply poor performance. For example, Tanaka found that TL was low over hard sand bottoms but ESL was high due to the number of multipaths that propagate to long ranges. The low TL more than compensates for the high ESL in this case.

Tanaka also compared his results with other methods of measuring ESL. He found that the assumption of a Gaussian distribution failed to work in certain conditions since the time spreading of the pulse was not decidedly non-Gaussian in shape. He also found that the Jones method of predicting ESL was highly dependent upon the size of the resolution cell chosen. By first using a 'best guess' resolution cell size that was too small to predict ESL, Tanaka found this best guess overestimated ESL by 2.4 dB. This method

of calculating ESL could be brought into agreement with Tanaka's more accurate depiction by selecting a different size resolution cell.

Adams (1997) continued the work in ESL for low frequency active sonar. He used the same FEPE model that Tanaka used, but his center frequency was at 250 Hz and he introduced a variable bathymetry. The geoacoustics and bathymetry were taken from the Tanner Bank area off San Diego. He found that ESL was greatly dependent on the time duration of the pulse. A 0.1 second CW pulse was found to have up to a 10 dB increase in ESL over a 5 second CW pulse. A longer pulse was also found to have less dependence on the geoacoustics of the bottom. He used a resolution cell method to process ESL. ESL was determined by finding the peak energy of a resolution cell in the output pulse and comparing that with the peak energy of a resolution cell in the compressed pulse. Due to overlap of the time spreading in a long pulse, a 5 second CW pulse was not deformed by the spreading caused by the multipath propagation. A long pulse was also less dependent on sediment factors for spreading loss. ESL remained at the same low level for the 5 second CW pulse over both a sand and silt/clay bottom. Using the resolution cell method, he also concluded that a Blackman windowed pulse has similar ESL levels of that of a CW pulse of the same time duration.

III. GEOACOUSTIC MODELING OF LONG AND ONSLOW BAYS

Long and Onslow Bays are located off the coast of the Carolinas close to the major Navy bases found along U.S. east coast. As seen in Figure 3, the two bays are divided by Cape Fear. The figure also shows that Long Bay was used as the site of the Focused Technology Experiment (FTE 96-2). Onslow Bay is being planned as the site for an underwater acoustic range. The areas are near the Gulf Stream, but most of the time the Gulf Stream is deflected from the Long Bay area by the Charleston Bump (Figure 4)(Kerr et al, 1997). The Gulf Stream starts to meander around this feature and can set up highly variable currents around this area. For this study, both areas have been modeled for evaluating the impact of ESL on different geoacoustic conditions.

A. ONSLOW BAY

Onslow Bay lent itself for studying the effects of a range dependent water column on ESL levels. In Onslow Bay, the water properties are highly influenced by the presence of upwelling zones. Upwelling can occur for a variety of reasons along the shelf break in this region. If the Gulf Stream moves offshore, cold water is allowed to be upwelled by wind forcing along the shelf break (Hofmann et al., 1981). The southerly winds that promote upwelling are predominately found in the summer months (Bucca et al., 1997). Another method of upwelling is the generation of warm core filaments that protrude from the Gulf Stream. Figure 5 shows an AVHRR SST imagery taken during FTE 96-2 around August 1996 (Bucca et al., 1997). The water temperature indicated by the green areas are approximately 14-15 °C. The Gulf Stream Front can be seen around the yellow areas which denote the 16-24 °C isotherms. The deep red maximum shows the center of the Gulf Stream at 28 °C. Gulf Stream filaments generally do not appear to break off to form an eddy, but they can persist from 6 days to 3 weeks (Atkinson et al., 1980). Figure 6 is a schematic illustration of the current pattern around a filament. The inside edge of the

filament has a cyclonic pattern that creates upwelling while the center of the filament contains an anticyclonic pattern that can cause down welling (Bucca et al., 1997). The filament pattern of alternating upwelling and downwelling cells traps a region of cold water around the shelf break. The trapped water then moves up and on to the continental shelf (Figure 6). The trapped water has been observed to occur every 14-40 days and account for about 20 percent of the water in Onslow Bay (Hofmann et al., 1981). The trapped layer of cold water rarely reaches the surface but does when there are upwelling events on the continental shelf that can force the water to the surface. The packets of trapped water vanish either when the shelf water mixes, when advection occurs, or when currents move the trapped water off the coast (Hofmann et al., 1981).

For this ESL study, the water column was modeled for a condition when a warm core filament was present and intruded into Onslow Bay. Data was taken from the Hofmann et al. (1981) paper. A map of the station numbers can be found in Figure 7. Figure 8 shows the temperature and salinity distributions for the central station numbers. Using the data found in this figure, an acoustic model was developed to study conditions between stations 41 and 43; later runs used stations 44-46. Temperature data was taken directly from Figure 8 (top), and salinity was assumed to be linear between the surface and bottom measurements. Figures 9 and 10 are a representation of the SSP determined by the Chen and Millero (1977) equation for sound speed for the above data. The sound speed profiles were used for the input to the FEPE modeling program. Table 1 indicates which runs in this study used multiple sound speed profiles.

As no geoacoustic data was available for Onslow Bay, the geoacoustic inputs for the FEPE model were taken from a Hamilton geoacoustic representation of the sand and silt/clay sediment types found in Area Foxtrot (Scanlon, 1995). Most of the runs use the highly reflective sand bottom; the silt/clay absorptive bottom was used for comparison with the sand bottom results. The sediment type was kept constant through the runs.

The bathymetric data along the modeled acoustic path was taken from Figure 8 (top). The depth of the water ranged from 25 to 37 m. Although achievable, this depth is

borderline for shallow water submarine operations, but the depth is ideal for implanting moored and bottom mines.

B. LONG BAY

The acoustic paths in Long Bay were chosen to follow the reconstructed ship tracks from a System Concept Validation test (SCV-97). An emphasis was placed on recreating the bottom type associated with these runs. The bathymetries along these paths were taken from two different charts. Figure 11 shows a large scale chart of the area with smoothed bathymetric features. A finer scale chart (Figure 12) reveals the complex pattern of the continental slope. The first few model runs use the smoothed bathymetry; a few runs were completed using the finer scale bathymetry to examine the requirement for an accurate bathymetry to predict ESL. The shelf area contains a wedge of sand lying over limestone layer (Figure 13), and this wedge was recreated in some runs to compare ESL dependence on changing bottom conditions. The sedimentary data was calculated from two nearby grab samples previously taken in preparation for FTE 96-2. Finally, the sound speed profile (Figure 14) of the water column was taken from a CTD cast taken during SCV-97 (Pasewark, 1997), and this profile is used for all the runs in Long Bay. The morning profile was taken during the SCV-97 experiment during the month of September. Previous sound speed profiles reveled that a significant time and space variability is present in the area. For comparison purposes, only one sound speed profile was used as shown in Table 1.

Table 1. Listing of the Runs Used for Both Onslow and Long Bays:

Number	Location	Bathymetry/Depth	Sound Speed Profile	Sediment
1	Onslow Bay	Downslope/25-33 m	Figure 9 Col. 1 2 3	Sand
2	Onslow Bay	Upslope/25-33 m	Figure 9 Col. 3 2 1	Sand
3	Onslow Bay	Downslope/25-33 m	Figure 9 Col. 1	Sand
4	Onslow Bay	Upslope/25-33 m	Figure 9 Col. 3	Sand
5	Onslow Bay	Constant/25 m	Figure 9 Col. 1 2 3	Sand
6	Onslow Bay	Constant/25 m	Figure 9 Col. 1	Sand
7	Onslow Bay	Downslope/33-37 m	Figure 10 Col. 1	Sand
8	Onslow Bay	Downslope/33-37 m	Figure 10 Col. 1 2 3	Sand
9	Onslow Bay	Upslope/33-37 m	Figure 10 Col. 3 2 1	Sand
10	Onslow Bay	Downslope/25-33 m	Figure 9 Col. 1	Silt/Clay
11	Onslow Bay	Downslope/25-33 m	Figure 9 Col. 1 2 3	Limestone
21	Long Bay	Downslope/125-225 m	Figure 14	Sand
22	Long Bay	Upslope/125-225 m	Figure 14	Sand
23	Long Bay	Upslope/75-125 m	Figure 14	Sand
24	Long Bay	Downslope/75-125 m	Figure 14	Sand
25	Long Bay	Varying/195-260 m	Figure 14	Sand
26	Long Bay	Varying/195-260 m	Figure 14	Sand
27	Long Bay	Upslope/125-225 m	Figure 14	Sand
28	Long Bay	Downslope/125-225 m	Figure 14	Sand
29	Long Bay	Downslope/125-225 m	Figure 14	Sand
30*	Long Bay	Downslope/125-225 m	Figure 14	Sand
31*	Long Bay	Downslope/125-225 m	Figure 14	Sand

* Run 30 has source depth at 55 m. Run 31 has source depth at 99 m.

IV. EVALUATION OF THE FINITE ELEMENT PARABOLIC EQUATION (FEPE) MODEL OUTPUT

A. FEPE_SYN

FEPE is a range-dependent, full wave model that calculates the propagation loss of a single frequency signal developed by Collins (1988). Range dependancies can be entered for any or all of the following: bathymetry, water sound speed profile, sediment sound speed profile, sediment density, and sediment attenuation. FEPE_SYN loops FEPE for multiple frequencies to create an ocean transfer function (OTF) of user-chosen bandwidth. The OTF can be used to evaluate propagation phenomena.

B. ESL CALCULATION

A Matlab program was written by the author to evaluate ESL using the OTF's saved in the FEPE_SYN output files. Figure 15 (a -f) shows the steps taken by the program to evaluate ESL. A time domain signal is chosen as an input pulse. A fast Fourier transform (FFT) is taken of the pulse to shift the pulse to the frequency domain. The OTF for a specific range and depth is retrieved from the output file. An output signal is created in the frequency domain by multiplying the frequency domain input pulse values with the OTF. An inverse fast Fourier transform (IFFT) constructs the output pulse in the time domain. A reduction value ('r') is calculated by comparing the energy in the output pulse with energy in the original pulse:

$$r = \frac{\sum (\text{Energy})_{\text{output pulse}}}{\sum (\text{Energy})_{\text{input pulse}}} \qquad (4)$$

The input pulse is multiplied by this value to create a reduced replica of the input pulse. This compressed pulse represents the ideal propagated pulse with TL taken into

consideration but with no time spreading. For purposes of evaluation, the peak of this pulse is considered the optimum achievable peak energy level. The maximum peak values of the compressed and time spread pulses are measured and compared to determine the one way ESL.

Figure 15 (a - h) demonstrates the steps taken to evaluate the mismatch loss (MML). Calculating MML is necessary to evaluate the effect of time spreading on continuous wave (CW) and frequency modulated (FM) pulses. The method discussed above to determine ESL in the time domain is applicable only for pulses containing one maximum peak. The frequency windowed pulses (Blackman, Hamming, triangle, etc.) meet this criterion. The multipath propagation takes energy from this single peak and spreads it into multiple peaks down range. Erroneous results appear when using this method to evaluate pulse patterns that have more than one maximum peak. A 100 millisecond CW pulse at 3500 Hz will have 350 peaks of the same size. As these peaks are spread in the time domain, constructive and destructive interference occurs. A replica pulse, with the same energy as the propagated pulse, may have peaks that are lower than the maximum peak of the propagated pulse. The calculated ESL value would be negative for this case. To determine the MML we continued from the preceding procedure, the input pulse is correlated with both the propagated pulse and the compressed pulse. The dB ratio of the maximum peaks are compared to determine the MML. Output from the auto correlation produces only one peak and any distortion of the signal will reduce the correlation. The compressed pulse will always have a higher peak value of correlation when compared with the time spread output pulse. This process is also closer to replica correlation processors uses in modern sonars such as the 53C or ALFS.

Figures 16, 17 and 18, respectively, show a Blackman pulse used for input, the output signal from 1 km to 20 km, and ESL calculated for the output signal. These figures illustrate ESL displays to be used in the analysis in the next sections.

C. MODEL INPUT AND OUTPUT

For the first part of the evaluation, certain model inputs were not varied unless noted later in the analysis. A Blackman pulse centered at 3500 Hz with a 201 Hz bandwidth was used to create a pulse in the time domain of approximately 0.02 seconds. With the step size in the frequency domain being 1 Hz, the Nyquist criterion requires that a minimum of 7200 frequency samples be taken for a maximum frequency of 3600 Hz to avoid aliasing. This requirement was met by using 16384 digital samples for the frequency and time domains. A 1 s window in the time domain creates a time step of 1/16384 second. A Gaussian starter parameter is chosen in FEPE, this requires that a small step size be selected to avoid aliasing. A resolution step size of 0.1 m for range was chosen which is less than 1/4 of the wave length, and the depth step size was reduced further to 0.05 m. The depth of the source was kept at 11 m for the 53C runs which is the approximate depth of this sonar array on a Spruance or Ticonderoga class hull. All the runs saved receiver depths at multiples of 11 m. Two additional runs were completed with source depths at 55 and 99 m. These depths were chosen for representation of an ALFS sonar below the layer. A tactical range of 20 km was chosen with reported results in 1000 m increments.

V. MODEL RESULTS

A. ANALYSIS OF ESL IN ONSLOW BAY

1. Analysis of Water Column on One Way ESL

Range independent propagation loss models are ineffective for predicting ESL in the Onslow and Long Bay environments. To analyze the influence of spatially varying sound speed profiles, the bathymetry was held constant. Runs 5 and 6 (Table 1) both had range independent bathymetries with a bottom depth at 25 m. The runs are identical except for additional SSP's water inserted into run 5 at 8 and 16 km. The effect of the varying water column is apparent, but small, beyond 8 km (Figure 19). The resulting difference remains small at less than 2 dB in one way ESL, and the moving average changes less by than 1 dB. The changes in SSP did not appreciably change the one way ESL results. [From this point, ESL will be used to mean one way ESL for brevity.]

Figure 20 shows ESL results for a downslope bathymetry from 25 to 33 m (runs 1 and 3 in Table 1). In all other inputs, runs 1 and 3 are identical to runs 5 and 6, respectively. The increased water depth results in an increase in ESL and a larger difference when the varying SSPs are introduced.

The up slope run, in the opposite direction of runs 5 and 6, is shown in Figure 21 for runs 2 and 4 (with the first sound speed change at 4 km). ESL is less for the up slope runs than for the downslope runs, as expected. A fluctuation occurs in the ESL difference above and below zero. It might be considered that an ESL oscillation is put into effect by the SSP changes. Examining runs having different water depths and SSPs disproves this hypothesis. Runs 7 and 8 (Table 1) are shown in Figure 22 for these situations. The bottom depth starts at 33 m and ends at 37 m. The mean value of the difference between the runs remains above zero from the first change at 4 km to the end of the runs at 20 km indicating that the varying SSP has increased the ESL (by 2 to 4 dB) for the entire path length.

When range-dependent SSPs are introduced, the variation in ESL levels can change rapidly. These increased variations would not be accurately predicted by a range independent simulation. Starting at 14 km in Figure 22, the difference in ESL values increase 5 dB in 2 km, while the next 2 km show a decrease of 5 dB. Figure 21 shows a similar situation where there is a 7 dB decrease in ESL followed by a 7 dB increase. These large differences can be attributed to the change in the propagating modes. As earlier discussed, ESL is caused by the spreading of the propagating energy due to the different group speed of each mode. When the SSP varies spatially, the interference pattern of the modes changes. This change in interference pattern is apparent in Figure 20(b) where the range independent case for sound speed exhibits a slowly increasing mean level for ESL. The dependent case(Figure 20(a)) exhibits a mean maximum at 13 km. This modal interference pattern change can be seen in all the figures discussed above.

The impact of a varying SSP environment on ESL was examined by comparing the variances of ESL and its mean over the 20 km acoustic path. This is shown in Table 2 for the sliding 5 point mean curve. For the scenarios depicted in Figures 19 and 20, a larger variance is experienced for the constant SSP case. The reverse is true for Figures 21 and 22 where more fluctuation is associated with the varying SSP case. This lack of consistency will limit the ability to accurately predict sonar performance on small scales (2 km), without using inversion techniques (Wilson et al., 1996) and measured reverberation data to deduce FEPE_SYN model inputs accurately.

Table 2. ESL Range Variability

Figure Number	Run Number	Bottom Depth	Variance (dB) From Mean Range Dependent SSP (a) Top Panel	Variance (dB) From Mean Range Independent SSP (b) Middle Panel
19	5 and 6	25	0.5478	0.6679
20	1 and 3	25-33	0.7683	1.3947
21	2 and 4	33-25	1.6023	1.2326
22	8 and 7	33-37	1.7946	0.5774

The time spreading of the transmitted pulse at ranges where large differences in ESL are seen in Figure 22(c) are shown in the next two figures. For the case of a constant SSP, Figure 23 shows that one primary modal group carries the energy with slight reductions over range. This relates to the relatively constant ESL level with range seen in Figure 22(b). The large changes in ESL imposed by changing the SSPs are shown in Figure 24. A large peak carries most of the energy at 14 km and 18 km (see Figure 22(a)) which relate to the low ESL levels, while varying SSPs have made a difference in the middle time series for 16 km. One SSP was entered at 14.5 km, and the large energy peak is no longer apparent at 16 km. This could be caused by destructive interference due to SSP changes in the water column that did not appear in the range independent case.

2. Analysis of the Impact of Bathymetry on ESL

Starting with range independent SSPs, Figure 25 compares the effect of an up slope to a down slope geometry. The top panel of Figure 25 shows that the downslope case exhibits an ESL trend that increases over range. The middle plot in Figure 25 shows the opposite effect for the up slope geometry as ESL decreases with decreasing water column depth. However, adding a varying SSP shows that this feature can reverse this trend, as shown in Figures 19 through 22. Figure 22 (two downslope cases with different SSPs) shows how dependent ESL is on each specific model input. The downslope case with range independent SSP (Figure 22(b)) shows that ESL decreases with increasing water depth. Introducing a range-dependent SSP (Figure 22(a)) does not reverse this trend; the ESL difference remains nearly steady between these two cases. Figures 26 and 27 illustrate two more runs with downslope versus up slope bathymetries. No definite conclusions can be drawn about the impact of upslope and downslope geometries on ESL, especially when the SSP varies along the acoustic path.

Differences in water depth for relatively shallow water regions (<40 m) may have a small effect on ESL. Figure 28 demonstrates this for two different runs that both have downslope geometries. The top and middle panels are for depth ranges of 25 - 33 m and 33 - 37 m, respectively. This comparison shows that ESL is slightly less (~2dB) for the

deeper water case. Can a greater difference in water depth significantly modify ESL values? Figure 29(a) shows a deeper water situation where the water depth initially starts at 75 m, and after deepening to 125 m at 5.5 km range, the bathymetry remains constant. The comparison is with run 6, a constant 25 m water depth with a constant SSP, shown Figure 29(b). The average difference in ESL for the entire path is less than 2 dB between the two cases. The ESL dependence on water depth for moderately deep water depths appears to be similar to that experienced when varying SSPs are introduced. In all the cases examined above, the water is relatively shallow, and many multipaths are present causing time stretching that is very sensitive to the local SSP and geoacoustic environment.

3. Analysis of Target Depth on ESL

All the runs in the previous discussions have the source and target in a possible surface duct at 11 m to simulate the performance of a 53C. In the following analysis, we examine one way propagation from an active source to a target. With the source remaining at 11 m, Figure 30 shows the target depth dependence of ESL. Run 3 has a downslope geometry and no change in water column SSP. This figure compares ESL for targets placed at 11 and 22 m. A large variation in ESL difference of plus and minus 4 dB is observed over the entire run. Figure 31 demonstrates the sensitivity of ESL between target depths when a spatially varying SSP is introduced. The fluctuation in ESL increases with over a 5 dB difference in ESL level noted at 13 km range. The range dependent SSP appears to increase the modal interference pattern and make ESL even more depth dependent. This can be seen by the time energy level at 13 km range (Figure 32) which shows the peak level, normalized to 1.0, is between two other peaks above 0.9. The majority of the energy is contained within these three peaks, and the ESL is large because of the time spreading of the energy over these three modal groups. When the target is at 22 m depth, Figure 33 shows only one major peak. At each depth within the water column, different modes carry the acoustic energy, and it is the time spreading of these modes that causes ESL. Thus, ESL is very target depth dependent and variable, but a

clear trend is not evident. Further examination of the impact of source and target depth on ESL will be addressed in the Long Bay analysis.

4. Analysis of the Impact of Sediment Properties on ESL

The effect of sediment properties on ESL is evaluated in this section. Run 3 uses a sediment 100 m thick containing high speed sand, while run 10 uses a sediment of similar thickness but a lower speed silt/clay sediment. As shown in Figure 34, the slow speed sediment produces a lower ESL consistently over these runs. However, low ESL is not tantamount to good sonar performance. In fact, just the opposite is true. The silt/clay sediment is not as efficient as the sand at refracting or reflecting the energy back into the water column. While ESL is lower over silt/clay sediment, TL is very high compared to the TL over hard sand sediment. Figures 35 and 36 show the difference in propagation of energy at the end of runs 3 and 10. The sand bottom has multiple modal groups carrying a significant amount of energy at the end of run 3. The silt/clay bottom only has a few significant modal groups, and the peak energy of the silt run is almost three times less than the sand run.

Figure 37 further illustrates the influence on the nature of the reflectivity of the sediment cover. The sand sediment of run 1 is changed to a hard limestone bottom with no sediment layer. In comparison to the sand sediment layer, the higher speed rock boundary reflects more energy back into the water column and produces higher ESL values. However, TL is relatively low for this latter case. The time domain energy plot (Figure 38 and 39) shows the difference between these two cases. The peak energy for the slower speed sand (Figure 38) is lower and the amount of energy in each of the modes is also lower. Another pattern occurs when examining these figures. The modal time pattern is similar between the hard sand and limestone sediment cases. Comparing the two time spread patterns in Figures 38 and 39, two peaks arrive after the main peak and three peaks arrive before the main peak. This similarity in patterns is not apparent in the previous example (Figure 36) with silt/clay sediments. The geoacoustical change from hard sand to silt/clay effects ESL significantly and the sediment properties limits the number of modal groups that propagate and the amount of energy that each of these modes can carry.

B. ANALYSIS OF ESL IN LONG BAY

1. Analysis of Water Depth on ESL

Runs 21 and 22 commence the Long Bay runs with water depths of 125 to 225 m. These runs have the same SSP but use an up slope (run 22) and downslope (run 21) bathymetry. Figure 40 shows low and decreasing ESL with increasing range beyond the first few thousand meters. The relatively deeper water (more than four times deeper) reduces the frequency of boundary interactions and hence reduces the degree of modal interference. In the next two runs (23 and 24), ESL is contrasted between up slope and downslope bathymetries. For the up slope case the bottom rises from 125 m to 75 m for the first 5.5 km and then remains constant thereafter. The reverse is true for the downslope simulation (Figure 41(b)). The ESL values initially start considerably lower for the up slope case, as expected, since the water column is deeper (125 m) at the start of run 23 compared to the constant 75 m depth of run 24 After multiple bottom interactions, the higher angle modes in both cases are attenuated , and the lower angle modes continue to propagate for the rest of the runs.

2. Analysis of the Impact of Target Depth on ESL

Initially, to examine the influence of source and target depth on ESL, the target depth will be changed to simulate a surface ship searching for a deep submerged target. Figure 42 shows how ESL changes when the target depth is lowered to 33 m for a source at 11 m. In processing several runs with different target depths, it was found that ESL is minimized when the target depth was at the same depth as the source. The largest increase in ESL level (5 dB difference) occurs as expected when the source and target are cross layer. It is apparent in the time spread pulses in Figure 43 that the target at 11 m has only one major mode (i.e., trapped in the mixed layer) that comprises a large amount of the total energy. The target at 33 m is cross layer and might be located in a shadow zone where is no direct path is present from the source (Figure 44). Since there may be no direct paths from the surface mixed layer to a cross layer target, acoustic propagation to

this location is by complex multipaths. The 33 m peak level is seven times lower than the 11 m peak level and five modal groups are located within 3 dB of the peak in Figure 44 (cross layer) compared to none in Figure 43 (in layer). The time spreading has increased ESL and created possible multipath returns.

Figure 45 shows ESL versus range for run 21, and the difference in ESL level between a target at 11 m and 55 m, respectively. As in the previous case, ESL is much greater for a deeper (cross layer) target. A similar situation, but for a 77 m deep target, is depicted in Figure 46. Both show ESL levels reduced below that for a 33 m target (i.e., immediately beneath the mixed layer) but the difference, relative to an 11 m target, is different at each depth. The time spread pulse at 9 km (Figure 47) for the 11 m target shows that a single group carries most of the energy. The time arrival structure for the 77 m target (Figure 48) has a higher peak level at 9 km than the 11 m target (Figure 47). ESL for a 99 m deep target shows about 2 dB higher ESL levels than for an 11 m target (Figure 49). Figures 50 - 53 show two more cases where the source is at 11 m and the target below the surface duct; the deeper target depth consistently demonstrates higher ESL. The results of this section show the need to employ a variable depth sonar, such as ALFS, to operate with the 53C and detect targets below the layer. ESL will be significantly lower for ALFS when searching for deep targets.

3. Analysis of the Impact of Source Depth on ESL

In the previous section, the source level was at 11 m to simulate a 53C sonar. With the availability of variable depth sonars deployed from a helicopter (e.g., ALFS) or surface ship, the active source can be placed below the surface duct. Different source depths will excite different modes in the water column. This pattern of interference will change the dominant modes and group speeds, and therefore the ESL levels can be expected to change as a function of source depth. In the previous sections, a general trend has been observed where lower ESL levels occur when the source and target are at the same depth. Deeper source depths positioned below the layer to detect deep targets reinforce these results. Figure 54 shows ESL for a source depth at 55 m for run 30

compared to run 21 with a source depth at 11 m. With both target depths at 11 m, a 6 dB average difference is noted for the last half of the run. When the source and target are both in the surface duct, acoustic energy can be trapped in the duct and essentially a single path (mode) propagates to the target reducing ESL. When the source is below the layer, there is no direct path to a shallow target. The path is made up of multi-modes, and large ESL levels occur. Figure 55 shows ESL for a source at 55 m depth with the target depths of 11 and 55 m. ESL is consistently lower when the source and target are at the same depth (55 m). The target that is cross layer from the source has ESL values that are 3 to 4 dB higher than the 55 m source-target combination. Figure 56 shows ESL for a source depth at 99 m with targets at 11 and 99 m. A 6 dB average increase in ESL results by the end of the run for a target in the duct and source deep below the layer. Figures 57 and 58 show the spread pulses for the targets at 99 and 11 m, respectively, for a 99 m source depth. The 99 m target in run 31 shows one main modal group carrying the energy through the entire 20 km range. The time arrival structure for the 11 m target shows a division of the energy into multiple modal groups. These multiple modal groups are each slowly reduced in level throughout the run as multipath effects reduce the energy in the run. By 20 km, little energy is left, and the energy is spread across a large interval of time. Figures 59 and 60 show the time arrival structure the end of this run. The energy for the target at 99 m is contained in one modal group. The target at 11 m has a peak energy level that is over 5 times less that at the deep target. The energy is spread across multiple modal groups, and this results in an ESL level of over 9 dB.

The ESL shown in Figure 61 is an exception to the above trend where ESL is greater for cross layer cases. The sources are placed at 55 m (Figure 61 a) and 11 m (Figure 61 b) with the target at 55 m. The ESL versus range difference (Figure 61 c) oscillates back and forth between plus and minus 2 dB even though this is a is cross layer simulation. Evidently, a source in the surface duct can ensonify the entire water column except for the immediate vicinity of the shadow zone. The same ESL effect is seen in Figure 62 for the source-target combinations of 11-99 m and 99-99 m, respectively. The ESL in Figure 62 actually shows a lower value at the beginning of the run for the 11 m

source (cross layer) than for the 99 m source. As emphasized previously, ESL and TL must be analyzed together with the remainder of the terms in the active sonar equation to make sonar performance predictions. In Figure 62 b, ESL may be decreasing with range because the TL is high and all the multipaths have been attenuated. The ESL results in this section simply imply that it is advantageous to ensonify both above and below the layer with active sonars such as the 53C and ALFS.

C. ESL DEPENDENCE ON PULSE SHAPE

In a complex way, ESL has been found to be dependent on the sediment properties, water depth, sound speed profile, source depth, and target depth as described in the previous sections. ESL is also dependent on the type of transmitted pulse. The previous discussions have used the Blackman pulse which was ideal for evaluating the output of FEPE_SYN. Run 1 (Table 1) was selected to compare the influence of ESL on the Blackman pulse compared to that for other pulse shapes. Figure 63 shows that there is a small increase in ESL when a boxcar pulse is used, especially within the first 6 km. This could be due to the higher time side lobe encountered when using the boxcar window due to its boxcar shape. The Hamming window also demonstrated a greater ESL level than the Blackman pulse as shown in Figure 64, but it was less than 1 dB for the entire run. Figure 65 shows that two similar windowed pulses, Hamming and Hanning, both exert a nearly similar effect on the ESL level.

Evaluation of the Blackman and boxcar pulse shapes over different run scenarios shows that the boxcar pulse has higher ESL levels than the Blackman pulse, in general. In run 8, the ESL values remain higher for the boxcar pulse as shown in Figure 66. In Figures 67 and 68, the Blackman and boxcar differences are virtually zero for the entire run because the ESL levels are very low for these runs. The pulse shape has an effect on ESL for situations where more multipath or modal interaction takes place. Windows that dampen the side lobes tend to have lower ESL values.

D. MML DEPENDENCE ON PULSE SHAPE

Previous results have been based on windowed pulses in the time domain having one maximum peak to calculate ESL to quantitatively define the time spreading effects. In sonar operations, other pulse shapes are transmitted. The continuous wave (CW), linear frequency modulated (LFM), and hyperbolic frequency modulated (HFM) pulses do not lend themselves to ESL calculations which are based on a peak level in the time arrival structure. Figure 69 shows the result of attempting an ESL calculation based on a peak maximum for an LFM pulse. Due to constructive interference, the peaks in the output pulse can be larger than the compressed pulse peaks at certain ranges, and this results in negative ESL calculations. This is due to the fact that the compressed pulse is a reduced replica of the transmitted pulse. To overcome this situation when the transmitted pulse is comprised of many peaks, a MML program was written by the author to correlate the output time series with the transmitted signal.

Figures 70-72 show that the Blackman pulse has the highest average MML values when compared to CW and LFM pulses of similar lengths. Average MML values were approximately 8 dB, 6 dB, and 4 dB for the Blackman, LFM, and CW pulses, respectively. The auto ambiguity functions can help to explain these results. Since the compressed pulse is a replica of the input pulse shape, the correlation of these two shapes will be identical in shape to the auto ambiguity function. MML is a measure of the distortion in correlation caused by time spreading. Since the Blackman pulse only has one peak, the auto ambiguity function can be approximated by a delta function, and therefore peak correlation for the Blackman pulse is highly susceptible to time spreading. The correlation of the LFM pulse can be related to its normalized auto ambiguity function of a rectangular envelope LFM (Ziomek, 1995):

$$|X_n(t,0)| = \begin{cases} (1-|t|/T)\,|\,\mathrm{sinc}\,(Dp * t\,(T-|t|)/\pi)\,|, & |t| <= T \\ 0, & |t| > T \end{cases} \qquad (5)$$

The correlation peak caused by the sinc function is narrow in time and hence useful in reducing range resolution problems, but lends the FM pulses to be susceptible to time spreading. Figure 73 shows that there is virtually no difference in ESL between the LFM and HFM pulses. The normalized auto correlation of the CW pulses returns a triangle which is the correlation of two rectangle functions (Ziomek, 1995):

$$|\mathbf{X}\,\mathbf{n}\,(t,0)| = \begin{cases} 1-|t|/T, & |t| <= T \\ 0, & |t| > T \end{cases} \qquad (6)$$

The base of the triangle acts as an integration of the time spreading and reduces the MML level. Figures 74 and 75 show that increasing the CW pulse length reduces the MML over the entire run. Figures 76 through 79 are graphical representations of the output of the correlator at the end of run 1. Figure 76 shows the auto correlation (unnormalized) of a CW pulse of 20 milliseconds. In Figure 77, the CW pulse shows the effects of the time spreading. There are two correlation peaks next to each other that degrade the peak correlation compared to the theoretical value shown in Figure 76. The MML is $10*\log_{10}(4.6/11.5) \approx -4$ dB for this case. A 100 millisecond pulse length shows less effect of time stretching due to the integration of energy. Figure 78 shows the auto correlation of a 100 ms CW pulse. Most of the time spreading has been absorbed into the main output time pulse in Figure 79 (MML = $10*\log_{10}(177/280) \approx -2$ dB).

Figures 80 and 81 show that MML for the LFM pulse is not affected by pulse duration. The next four Figures (82-85) show that the correlation of the compressed and output time series are similar for the 20 millisecond and 100 millisecond pulse length. Because of the large peak in the auto correlation functions for FM pulses, time spreading has a large effect in reducing overall correlation (MML) regardless of the pulse length.

E. INITIAL RAMP INCREASE IN ESL AT SHORT RANGES

Tanaka (1996) noted that ESL values increased in the first 1600 m of range and then remained nearly constant, but often with a slow oscillation or trend. For the

environments in this analysis, the ramp up of ESL was found to occur in the first 1000 m. Table 3 lists the ranges where the average ESL values over 250 m first begin to decrease. This tended to be a good measure of the critical range for the initial increase in ESL.

Table 3. Ranges of Initial Ramp Increase in ESL

Run Number	Distance (m) to Critical Range 11 Meters - Target Depth	Distance (m) to Critical Range 22 Meters - Target Depth
1	300	550
2	400	350
3	300	550
4	350	350
7	400	300
8	400	300
9	250	250
10	250	250
11	300	550
21	800	(33 m target) 150
22	600	
23	400	
24	400	
25	600	
26	650	
30	150	(55 m target) 400
31	200	(99 m target) 350

The range to the critical depth was dependent on the run environment. The average critical range for the Onslow Bay runs with source and target at 11 m were the shortest (268 m). Figure 86 shows that ESL increases rapidly for a 22 m target depth compared to the gradual ramp up for ESL for the 11 m target depth case. Figure 87 shows that the initial ramp up of ESL occurs for both silt-clay and sand sediments, even though the overall ESL values are less for silt-clay. Figures 88 and 89 compare the ESL increase for various source-target combinations. In Figure 88(d) ESL ramps up quickly to 6 dB but begins to decrease as higher angle rays start to interact with the bottom. The quickest ramp ups in these figures are for source-target combinations that cross the layer. This could again be due to the fact that for cross layer situations no direct path propagation is possible and only multiplaths contribute energy to the target.

VI. CONCLUSIONS AND RECOMMENDATIONS

Model inputs were chosen from Onslow and Long Bays off the Carolina coast to study the impact of various factors on ESL. Tactical frequencies were used to simulate the 53C and ALFS sonar systems. Sediment properties, SSPs, water depth, source depth, and target depth were modified for comparison of transmitted ESL. The input pulses were also evaluated. For box shape pulses (CW and FM), MML was used to determine the impact of time spreading on the transmitted pulse shape.

ESL is dependent on how the sound speed profile (SSP) varies along the acoustic path, but the effect can not be assessed accurately without first measuring the highly variable spatial SSP in shallow water. By itself, the dependence on SSP is small with fluctuations varying between plus and minus 2 dB. However, when coupled with a range-dependent bathymetry, the effect of a variable SSP increases the influence of ESL, increasing ESL fluctuations to almost plus or minus 5 dB.

An absorptive bottom attenuates most of the higher mode energy and reduces ESL. Even though ESL is lower, the transmission loss is higher, and there is less energy that can be gained by signal processing. A reflective bottom creates many multipaths resulting in significant time spreading of the signal, which results in higher ESL values. This does not necessarily cause degradation in performance because TL is low and signal processing techniques can regain the ESL.

ESL is dependent on source and target depths. The surface layer effectively traps acoustic energy and can propagate a single modal group, or ray bundle of energy, which reduces transmission loss and ESL. The largest ESL losses are for those scenarios where the source and target are cross layer. Generally the lowest ESL values are found for the source and target located at the same depth.

Initial spreading of the pulse causes ESL to increase rapidly within 1000 yards. There is some dependency of this critical range on the source and target depths. A

comparison of time spreading with mode theory could lead to an explanation of why ESL increases faster for some environments.

Mismatch loss is a technique used to quantify the degree of ESL for signals having multiple peaks such as CW or FM pulses. MML was found to be dependent on the pulse shape with CW pulses showing lower MML than LFM/HFM pulses. A longer CW pulse integrates the energy that has been spread and reduces MML. FM pulse duration has little effect on MML. With HFM and LFM pulses demonstrating similar MML values for equal pulse durations. Higher order statistics and coherent processing could be studied for later use in evaluating correlation techniques that reduce ESL. Further study should focus on the advanced signal processing techniques, such as matched field processing and inverse beam forming, that could regain the energy lost to ESL. Other signal processing areas of interest would be the effect of noise limited environment on the recovery of ESL energy.

APPENDIX A

FIGURES

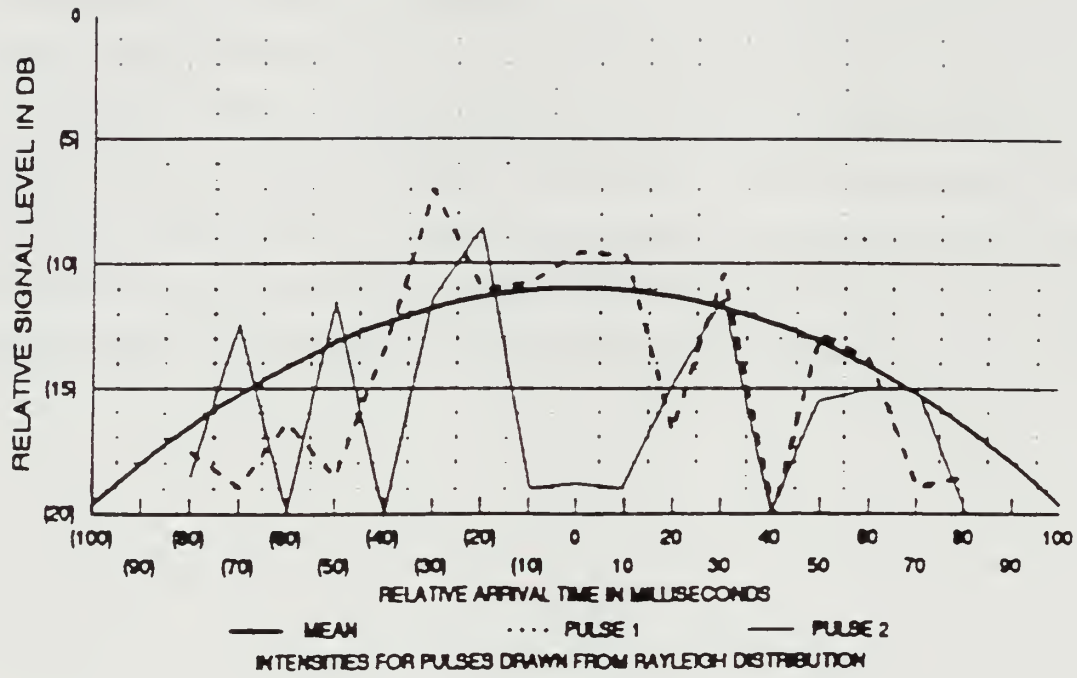

Figure 1. The comparison of a Gaussian spreading of mean intensities versus Rayleigh modeled spreading (from Bell, 1989).

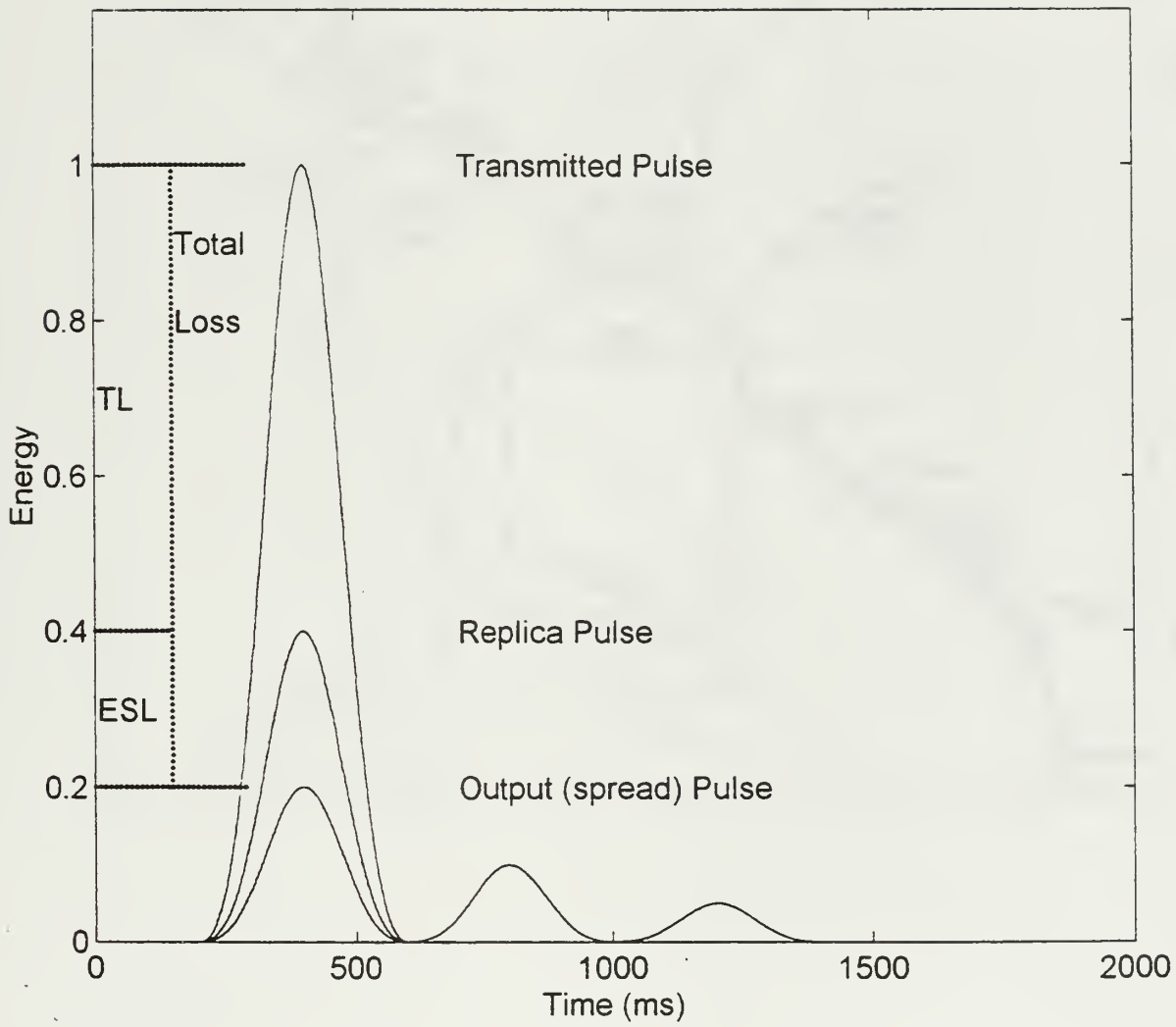

Figure 2. ESL visually defined by comparing the output (spread) pulse and the compress pulse. Power (or acoustic energy) of the pulses in the time domain.

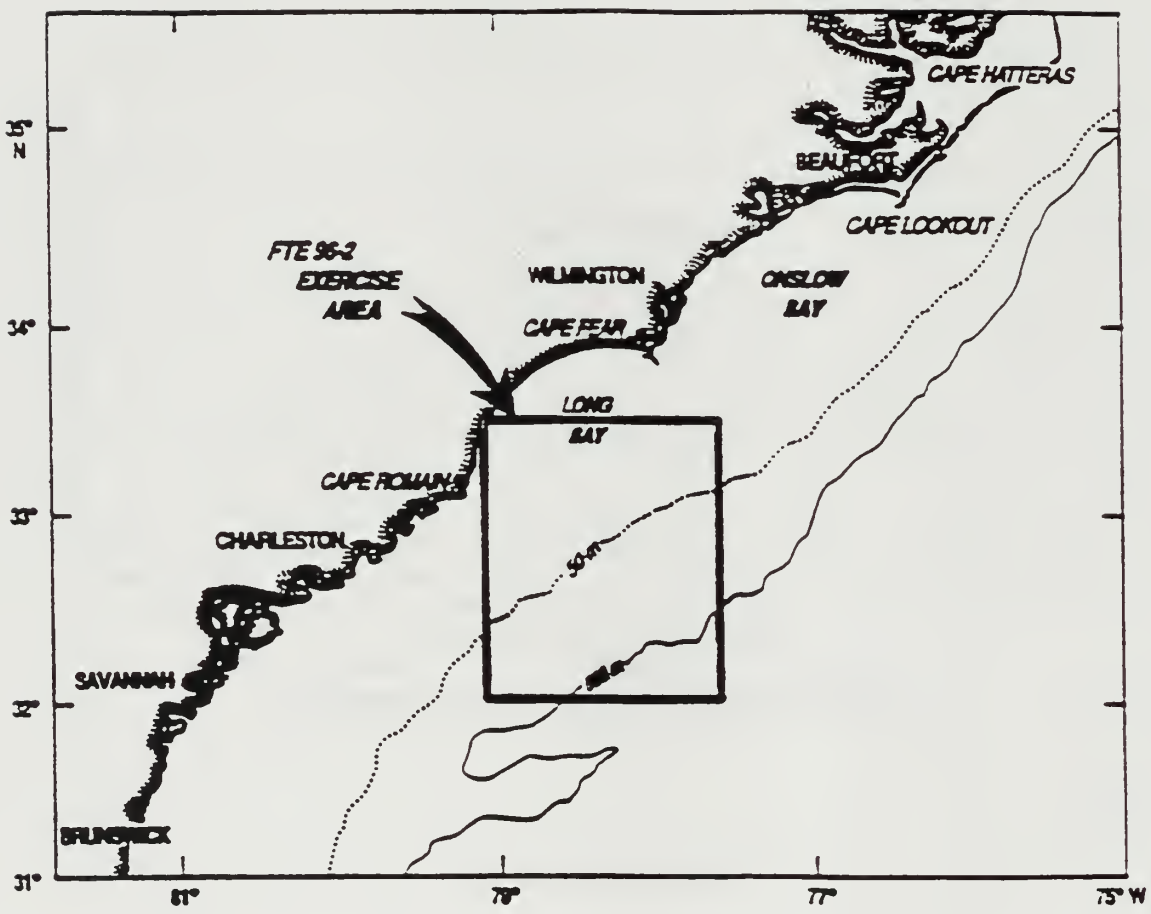

Figure 3. A map showing the area of interest. Long Bay was used for FTE 96-2, and Onslow Bay is planed for future exercises (from Soukup and Ogden, 1997).

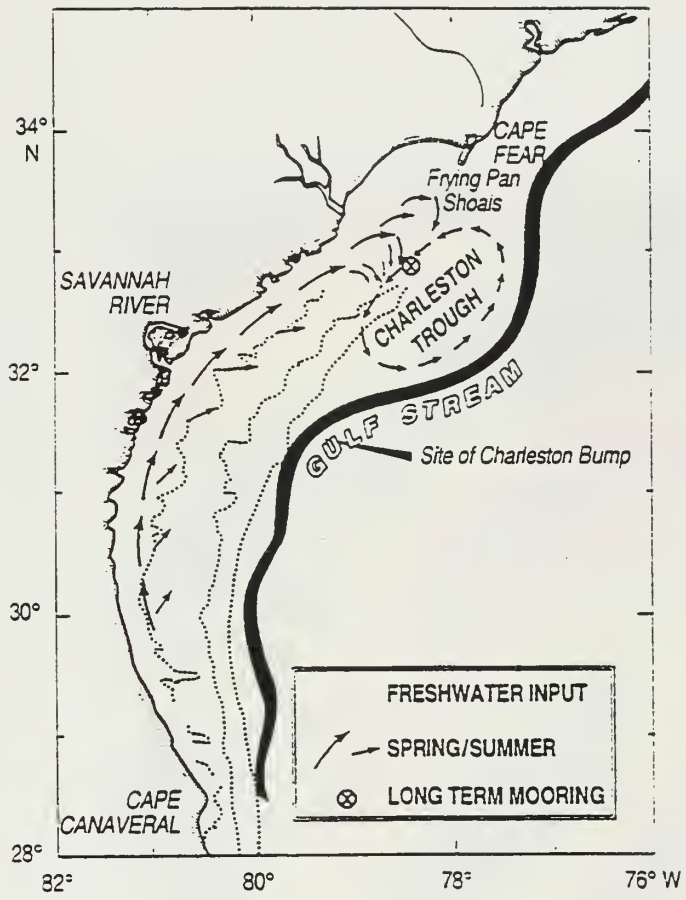

Figure 4. An illustration showing the location of the Gulf Stream, Charleston Trough, and path of coastal currents (from Kerr et al., 1997).

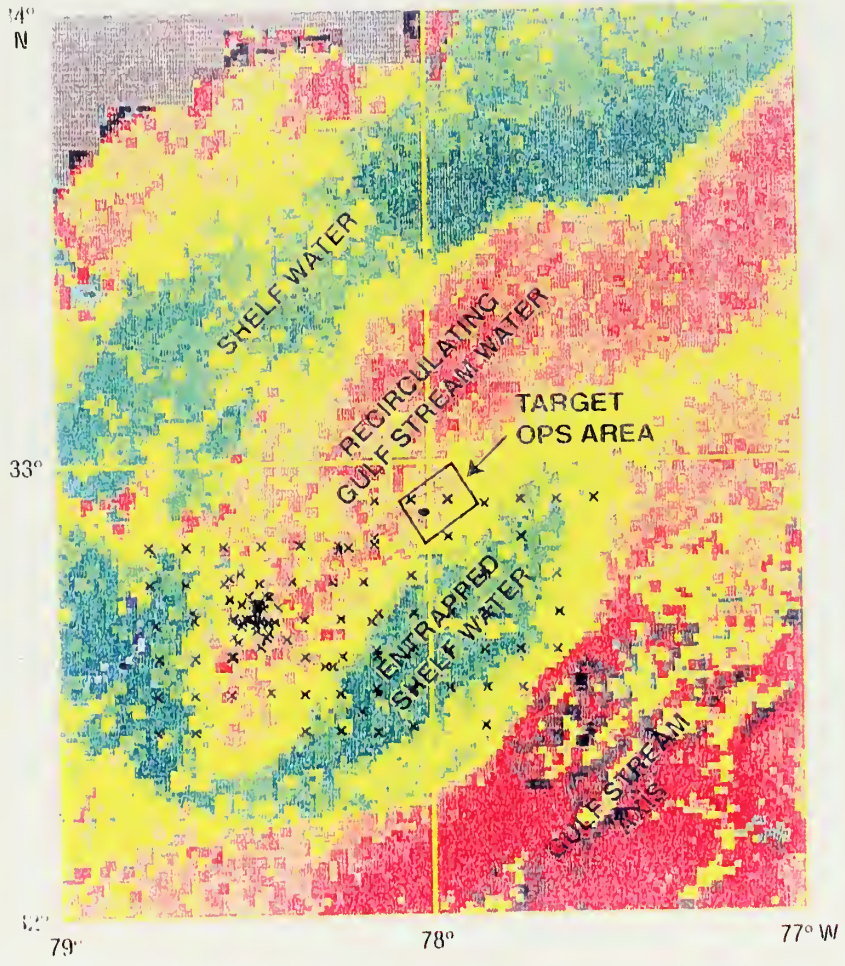

Figure 5. An AVHRR image, taken during FTE 96-2, showing the surface temperatures of a Gulf Stream filament (from Bucca and Fulford, 1997)

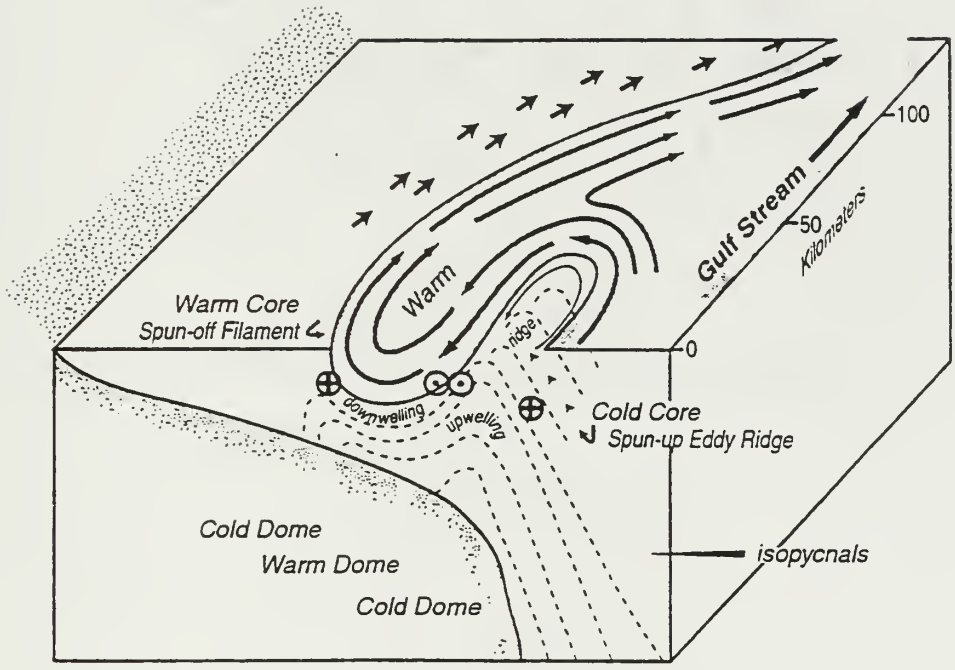

Figure 6. A conceptual diagram of a Gulf Stream filament based on AVHRR imagery and current meter moorings (from Bucca and Fulford, 1997).

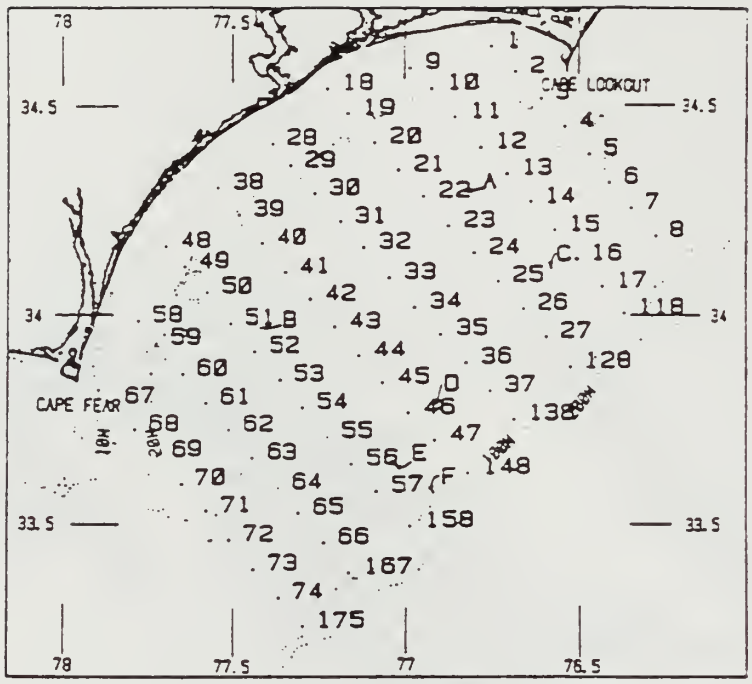

Figure 7. A map of station numbers in Onslow Bay (from Atkinson et al., 1980).

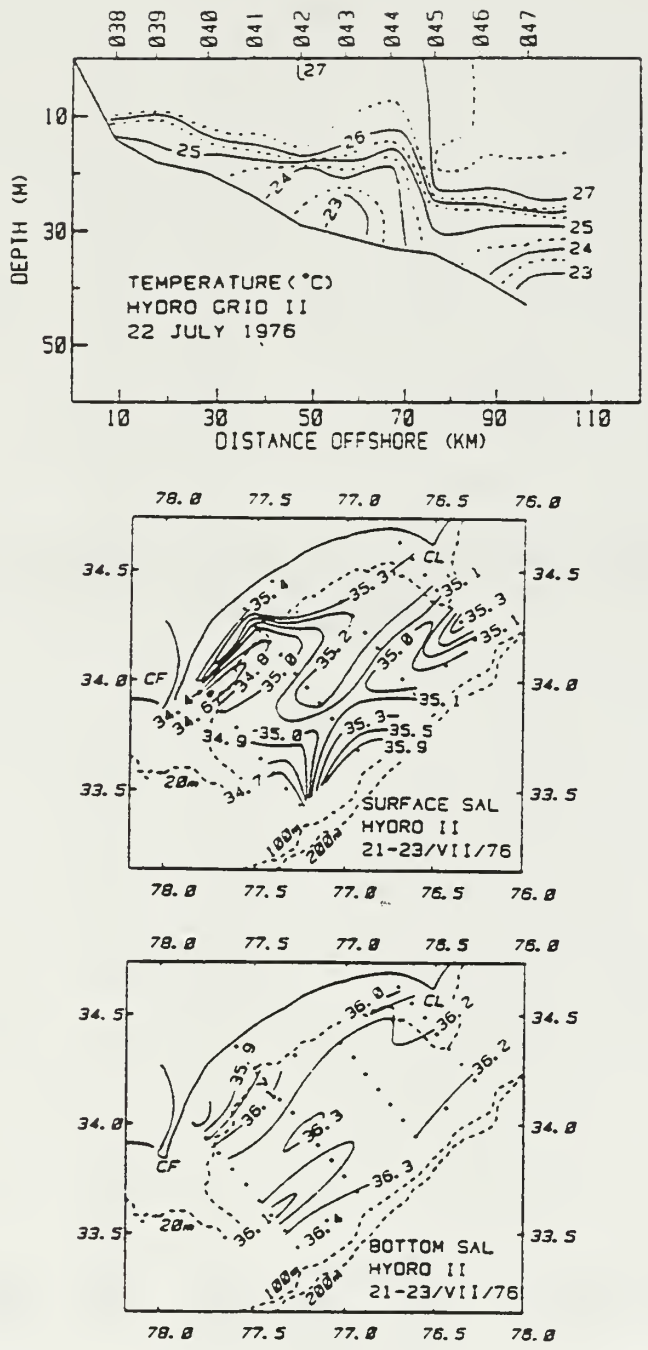

Figure 8. Top: Temperature profile showing trapped coastal water on the continental shelf of Onslow Bay. Middle: A map displaying surface salinity distributions. Bottom: A map displaying bottom salinity distributions. The data was taken during 21-23 July 1976 (from Hofmann et al., 1981)

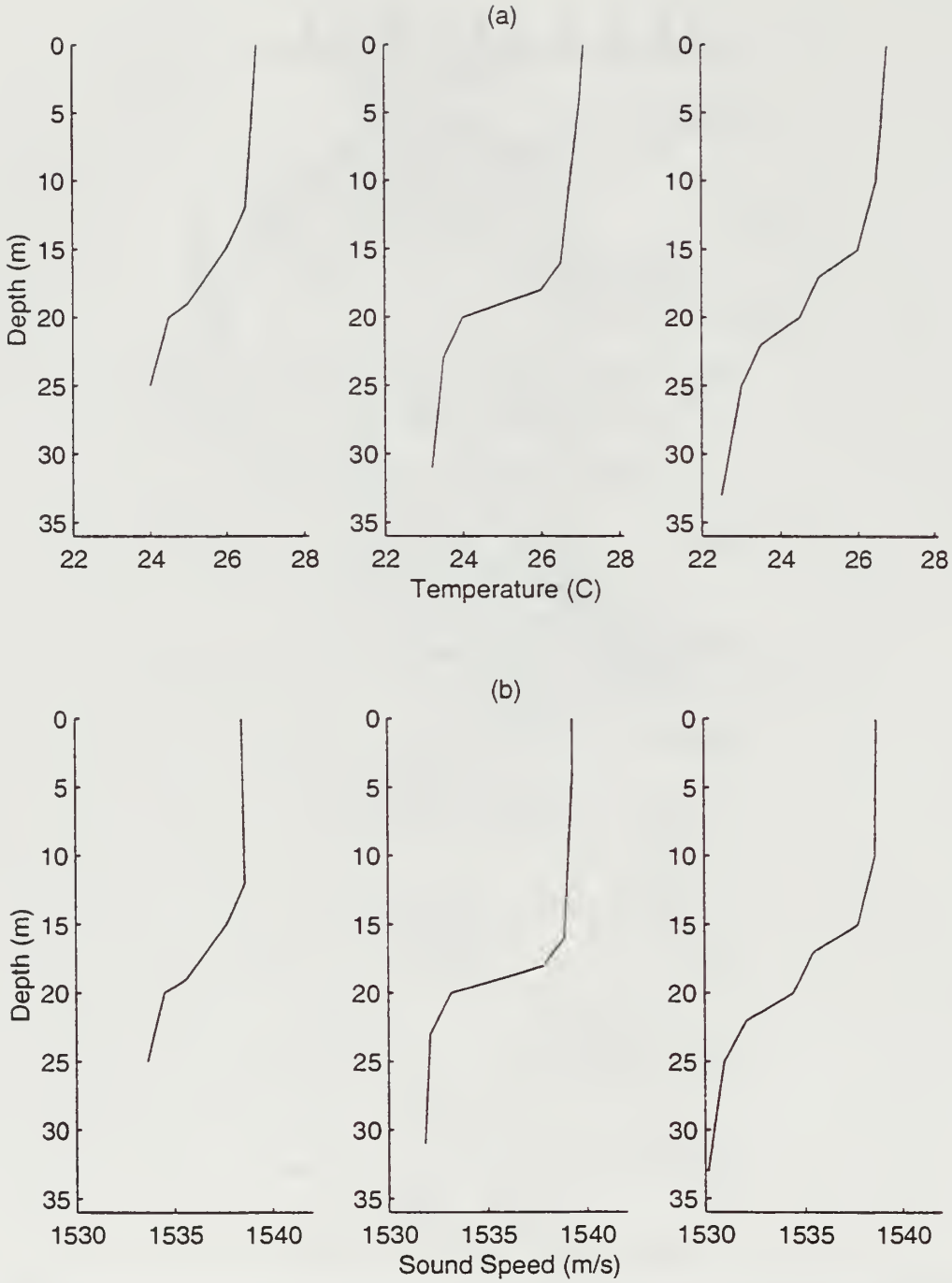

Figure 9. (a) Temperature cross section based on Hydro II data. (b) Sound speed profiles derived from the respective temperature profiles and salinity measurements. The columns, from left to right, represent Onslow Bay stations 41, 42, and 43.

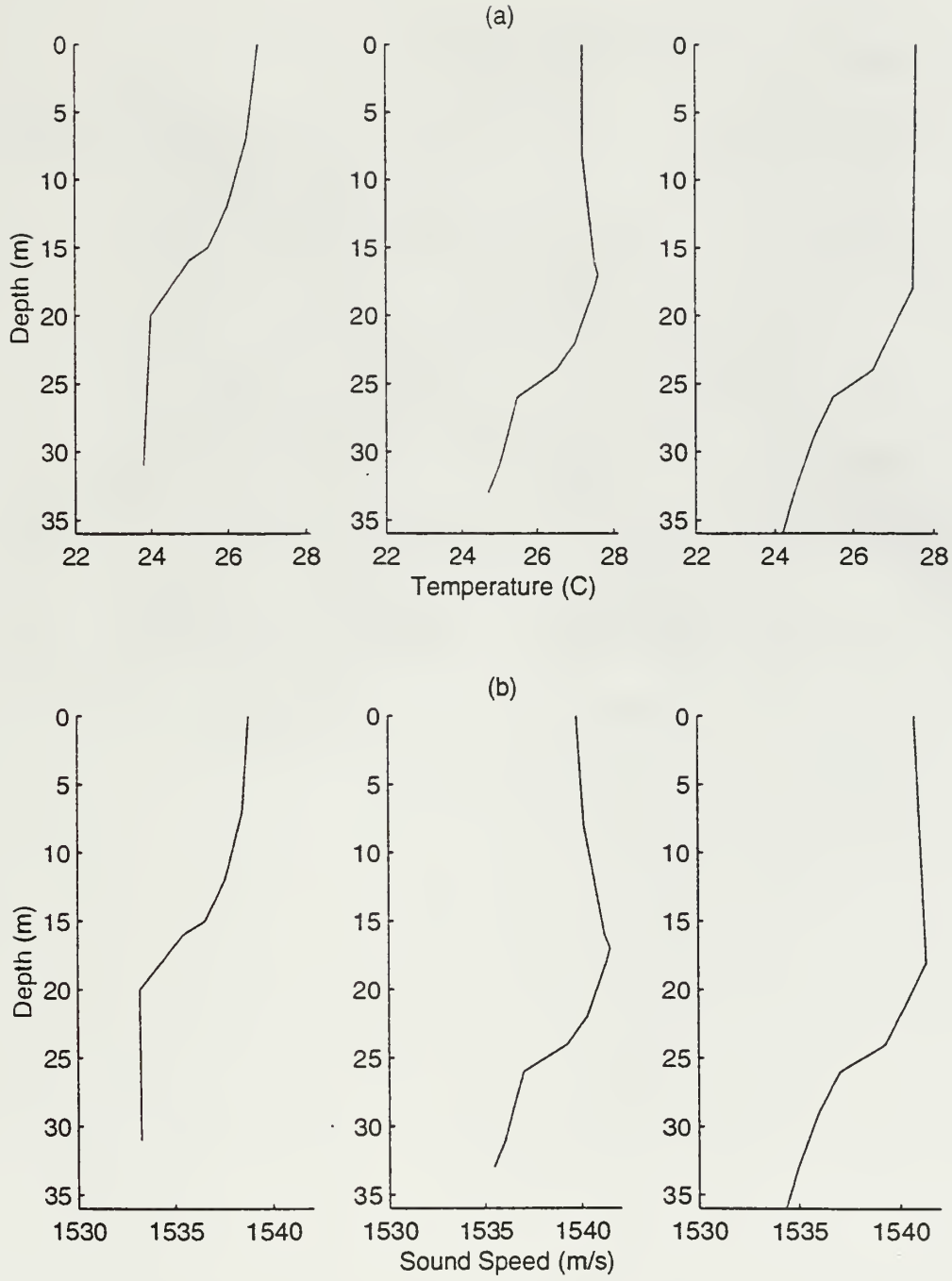

Figure 10. (a) Temperature cross section based on Hydro II data. (b) Sound speed profiles derived from the respective temperature profiles and the salinity measurements. The columns, from left to right, represent Onslow Bay stations 44, 45, and 46.

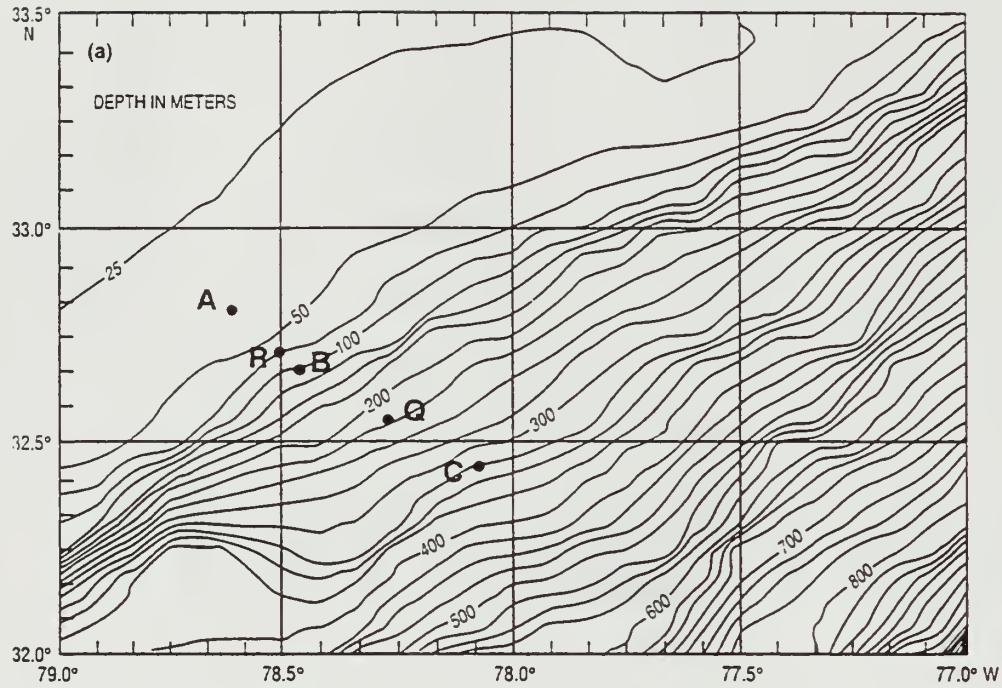

Figure 11. A chart displaying bathymetry in the planed area of FTE 96-2(from Kerr et al., 1997).

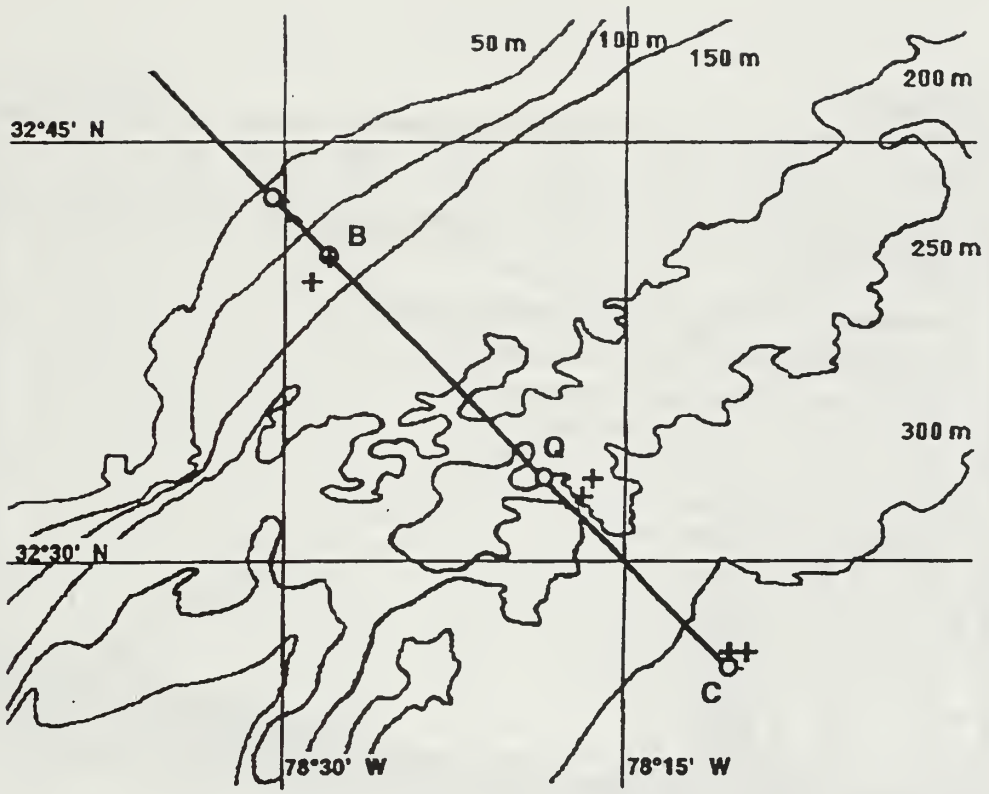

Figure 12. A large scale chart showing the same area as figure 12(from Soukup and Ogden, 1997).

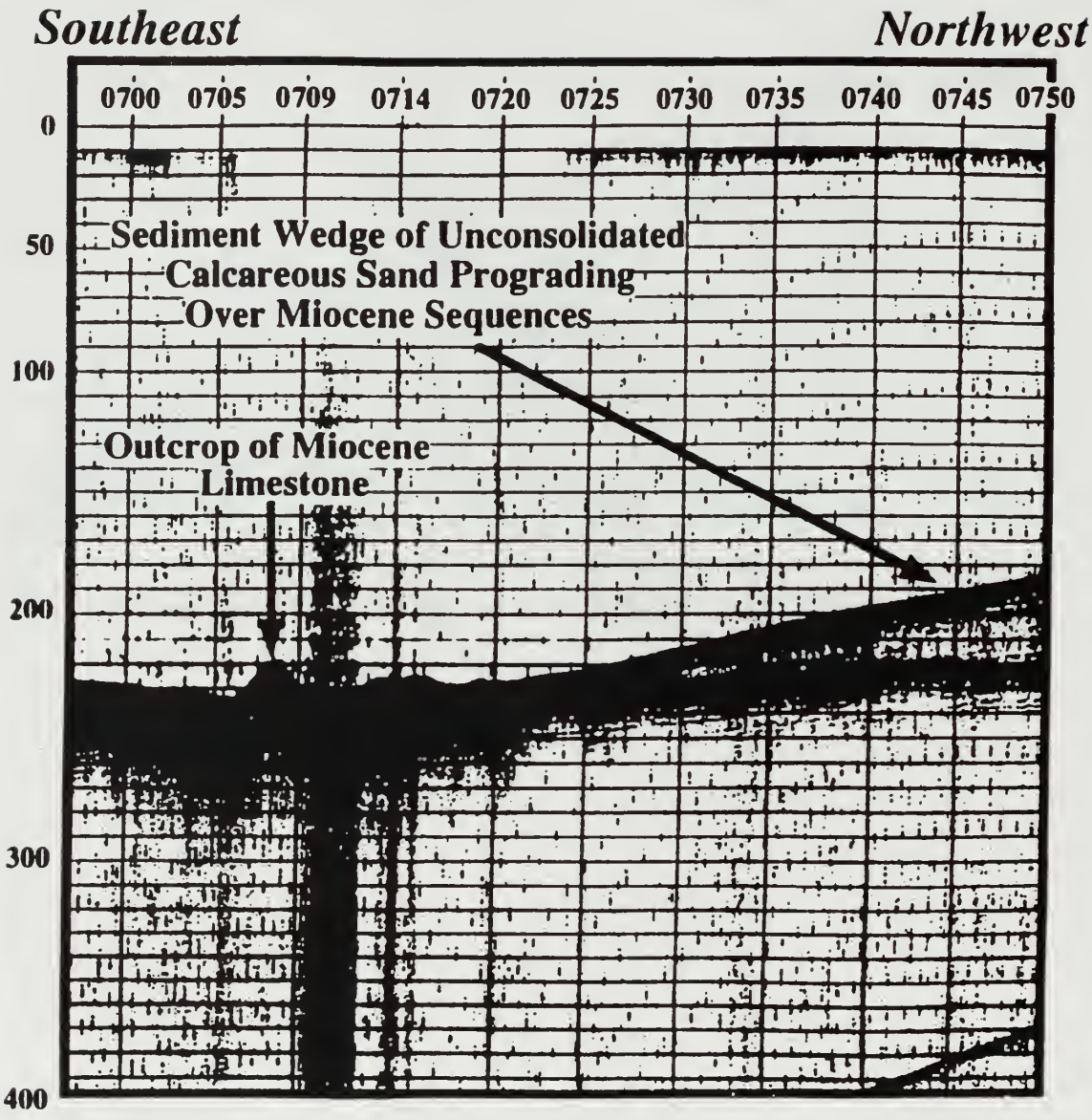

Figure 13. A graph of seismic data showing a sediment wedge in Long Bay (from Soukup and Ogden, 1997).

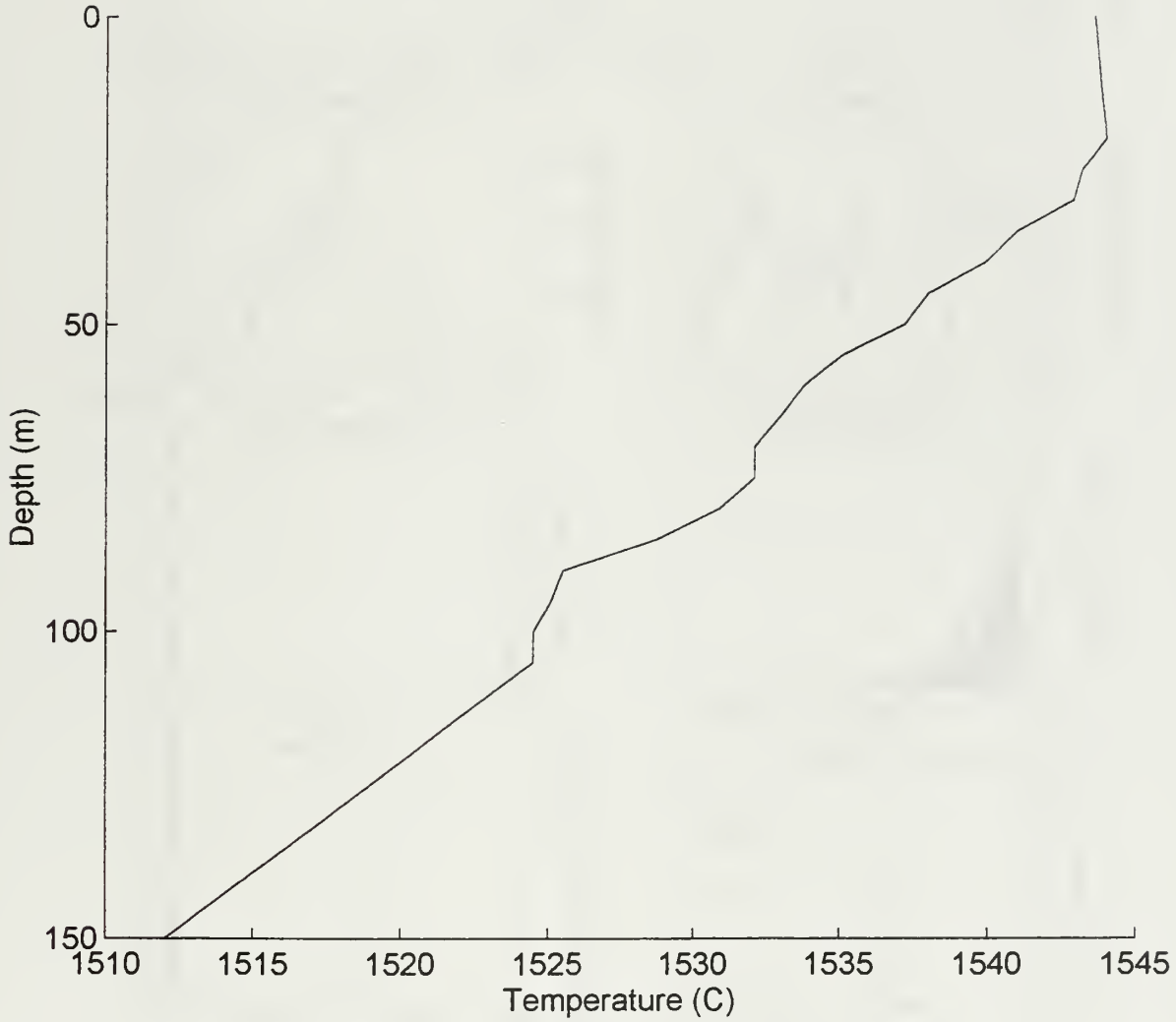

Figure 14. A sound speed profile of Long Bay taken from FTE 96-2 data.

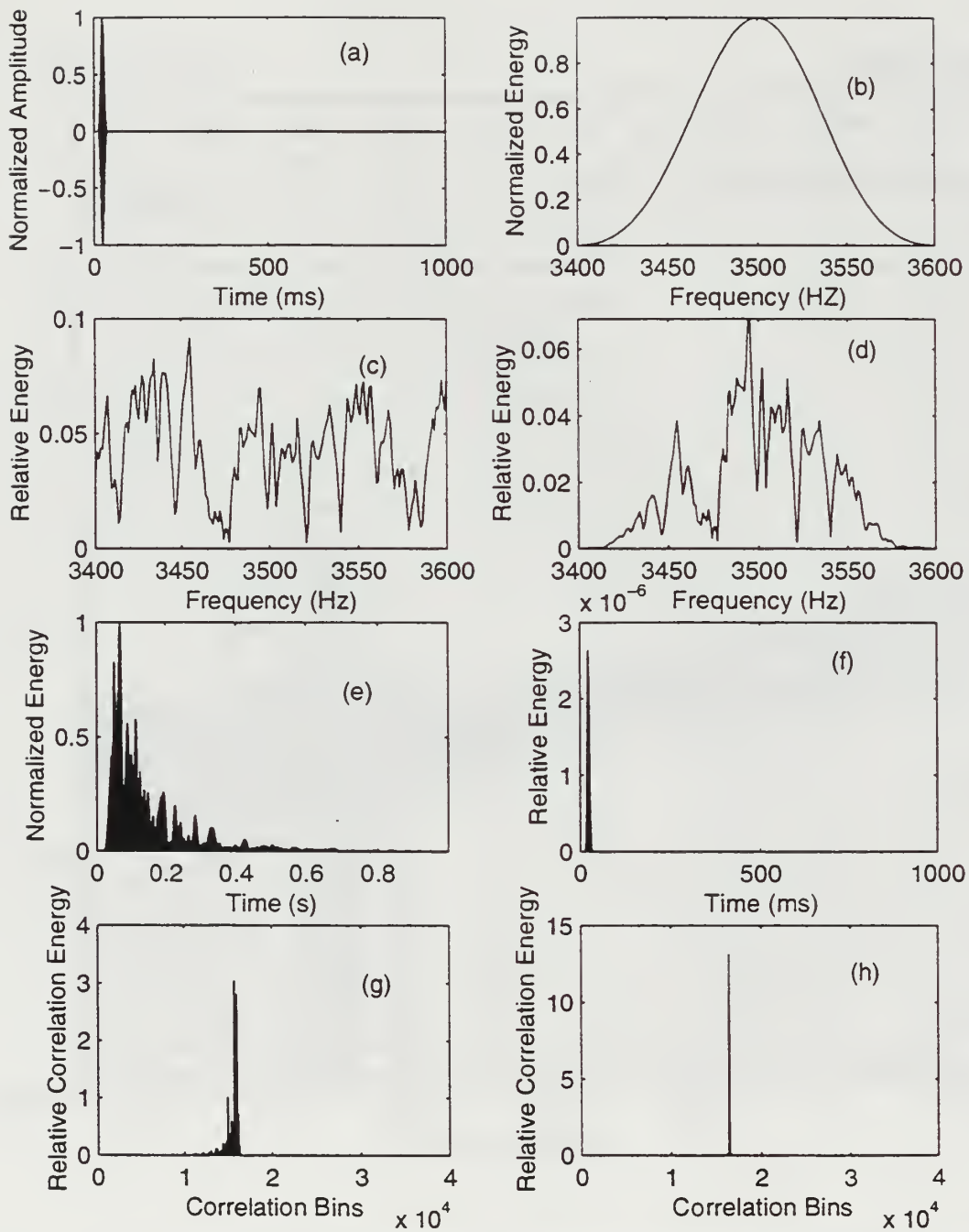

Figure 15. A graphical representation displaying the method of calculating ESL and MML. (a) input pulse in the time domain (b) input pulse in the frequency domain (c) ocean transfer function from FEPE_SYN (d) output pulse in the frequency domain derived from multiplying b*c (e) output pulse in the time domain (f) compressed pulse in the time domain (g) result of correlating a with e. (h) result of correlating a with h

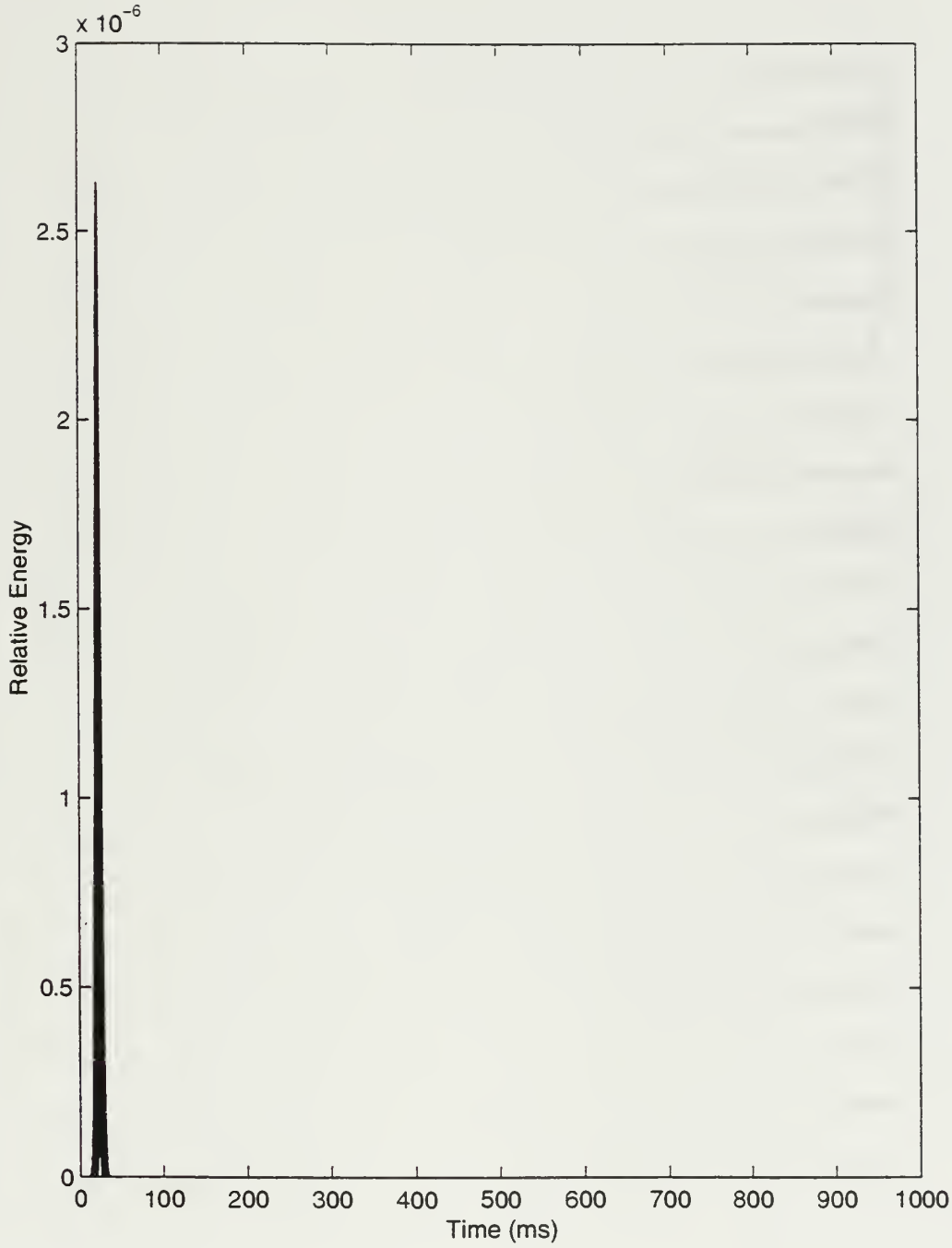

Figure 16. The Blackman pulse shown in the time domain.

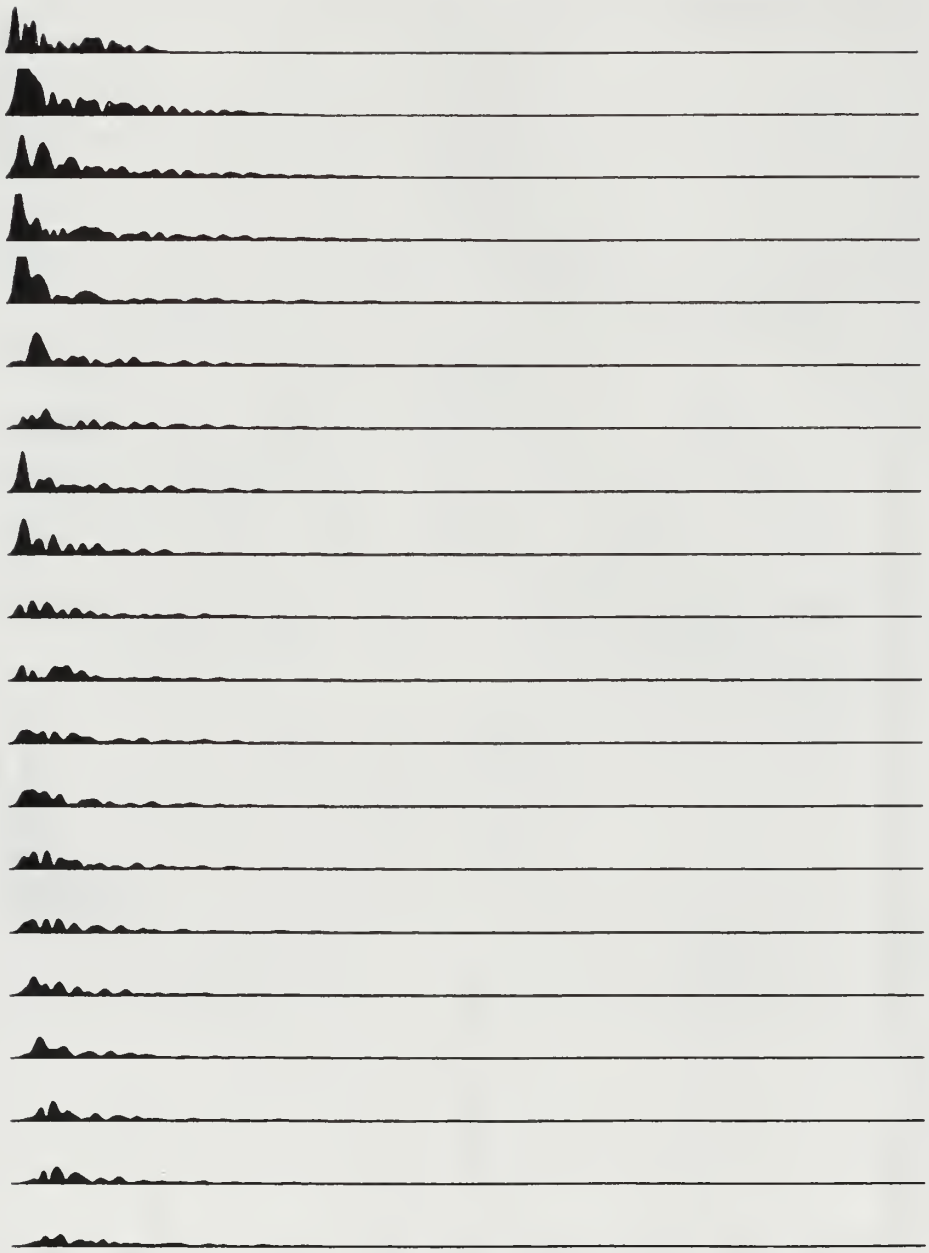

Figure 17. A plot of output pulses in the time domain. The figure starts with energy level at 1 km range at the top to 20 km at the bottom in increments of 1 km per graph. These pulses were derived from the OTF from run 1.

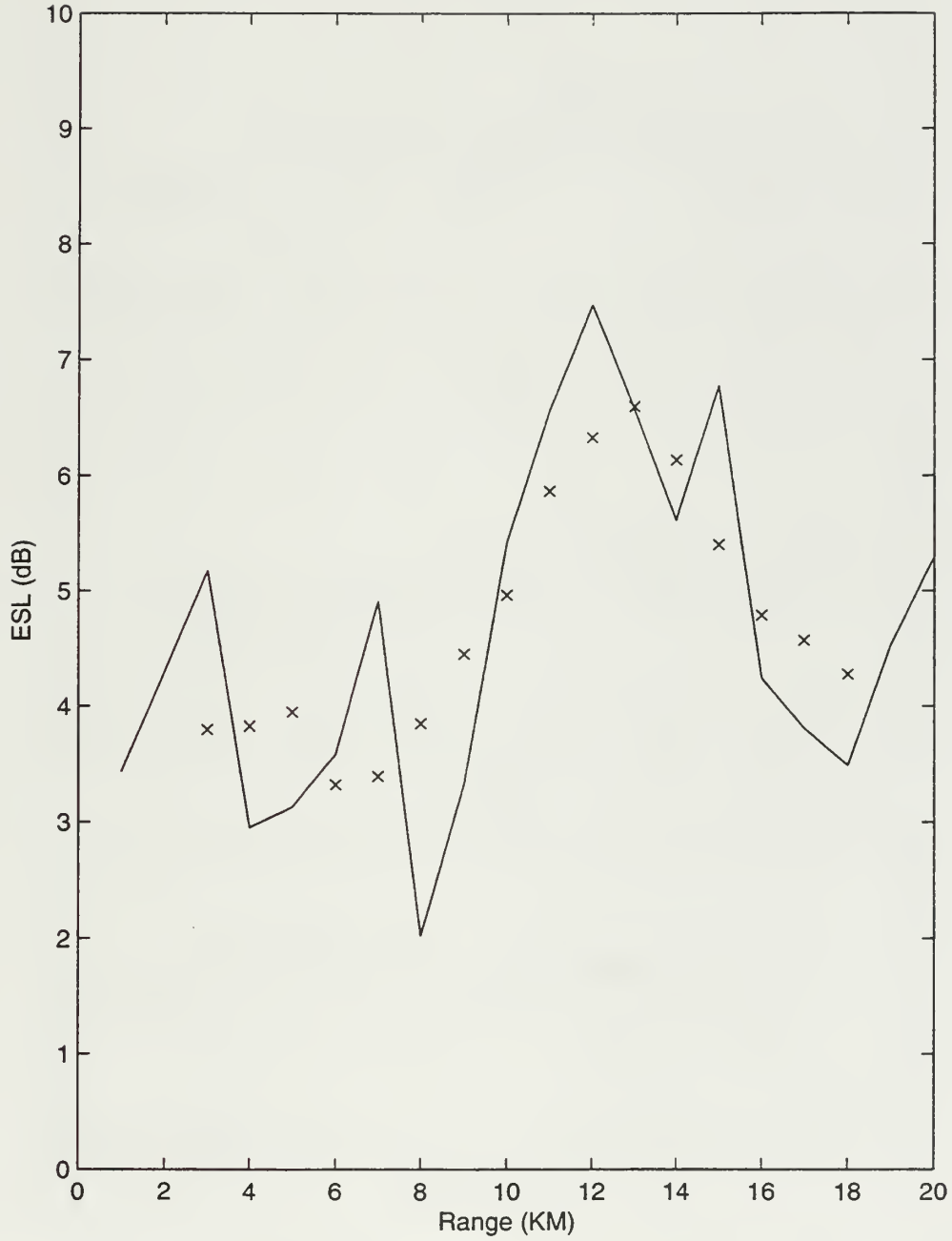

Figure 18. A plot showing the ESL of run 1. The solid line represents ESL at each range. The X's represent a 5-point moving ESL average.

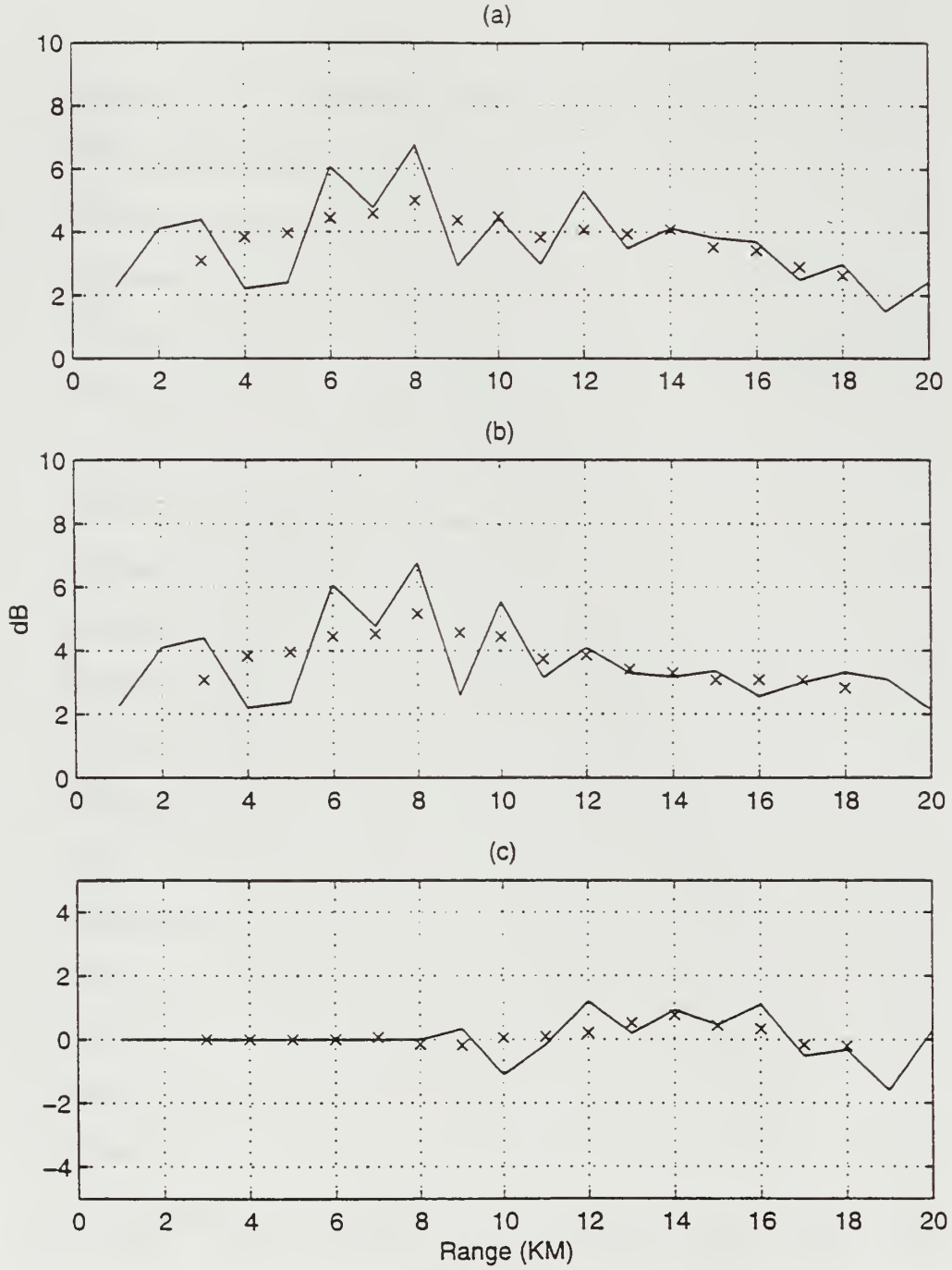

Figure 19. (a) ESL plotted for run 5. (b) ESL plotted for run 6. (c) The ESL difference between run 5 and run 6. The solid line represents ESL at each separate range increment. The X's represent a 5-point moving ESL average.

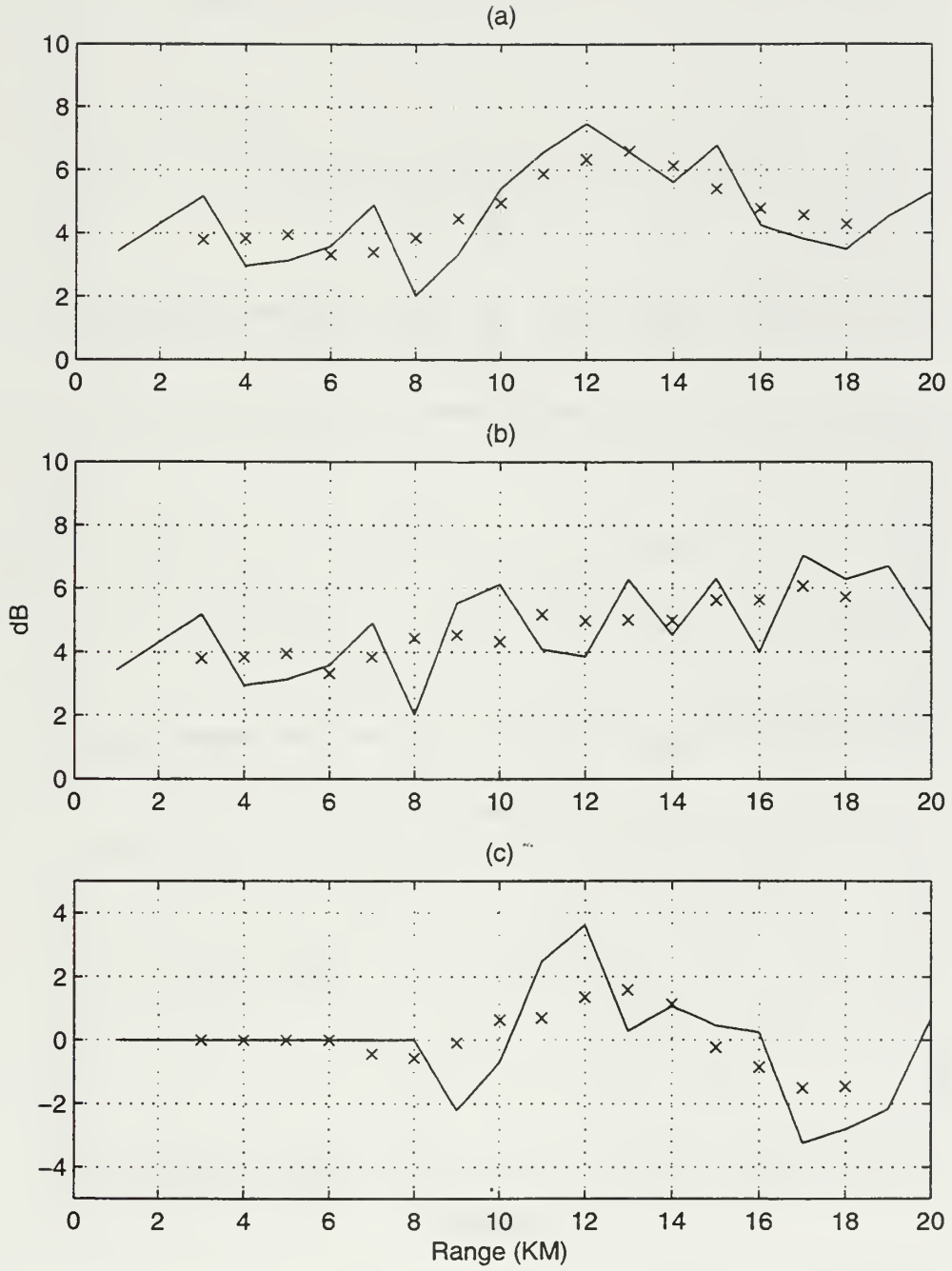

Figure 20. (a) ESL plotted for run 1. (b) ESL plotted for run 3. (c) The ESL difference between run 1 and run 3. (Graph properties as before)

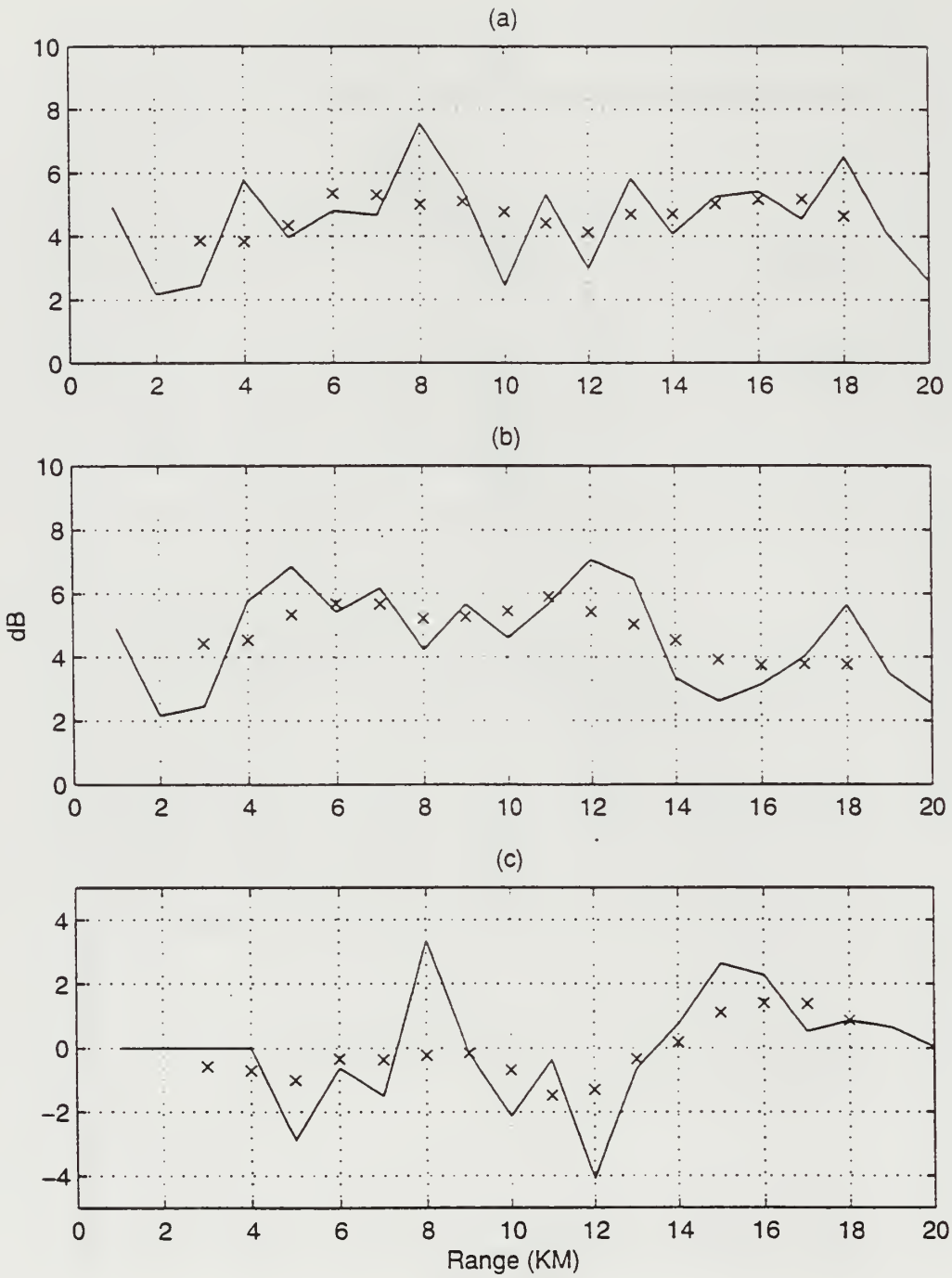

Figure 21. (a) ESL plotted for run 2. (b) ESL plotted for run 4. (c) The ESL difference between run 2 and run 4. (Graph properties as before)

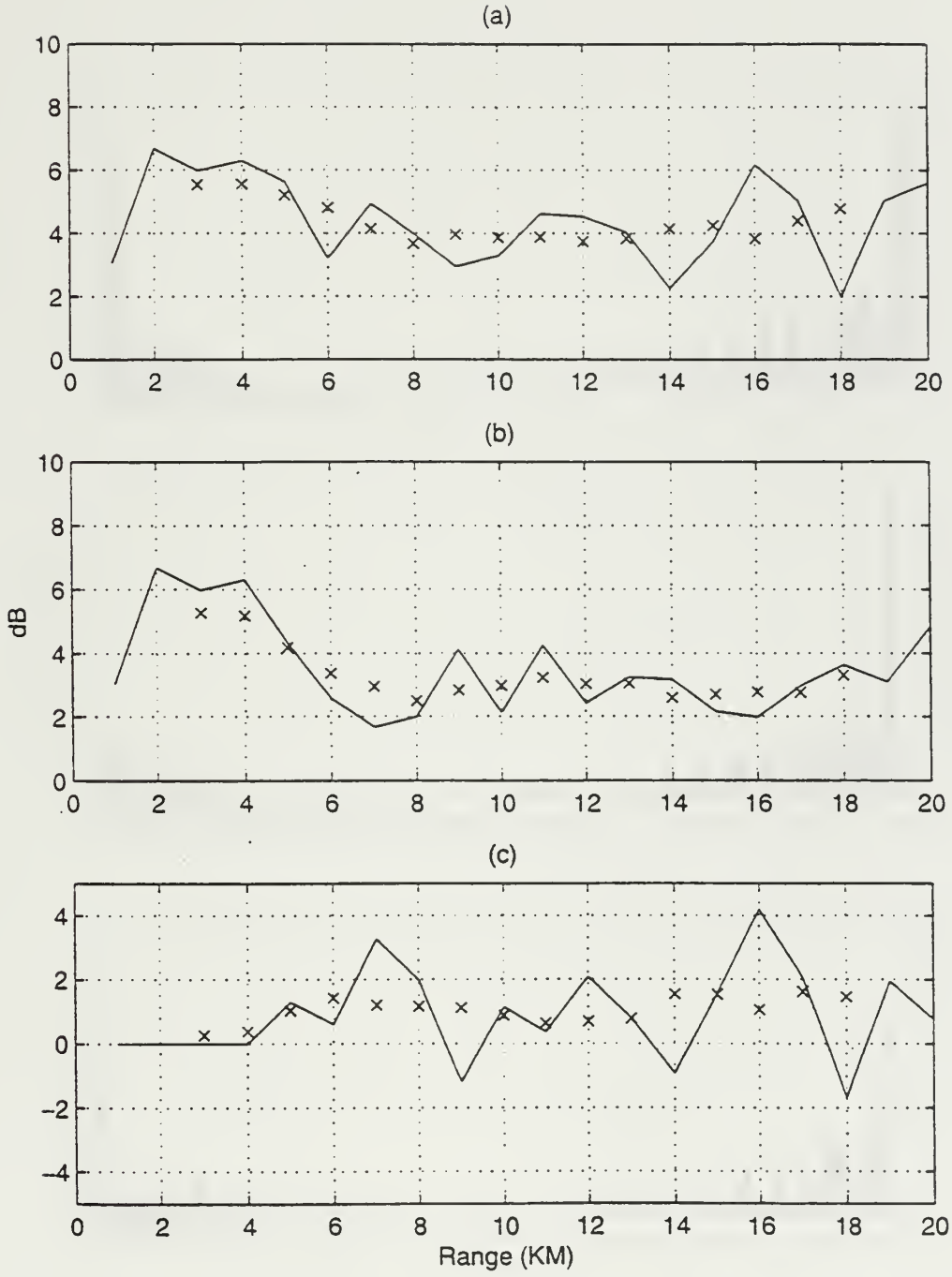

Figure 22. (a) ESL plotted for run 8. (b) ESL plotted for run 7. (c) The ESL difference between run 8 and run 7. (Graph properties as before)

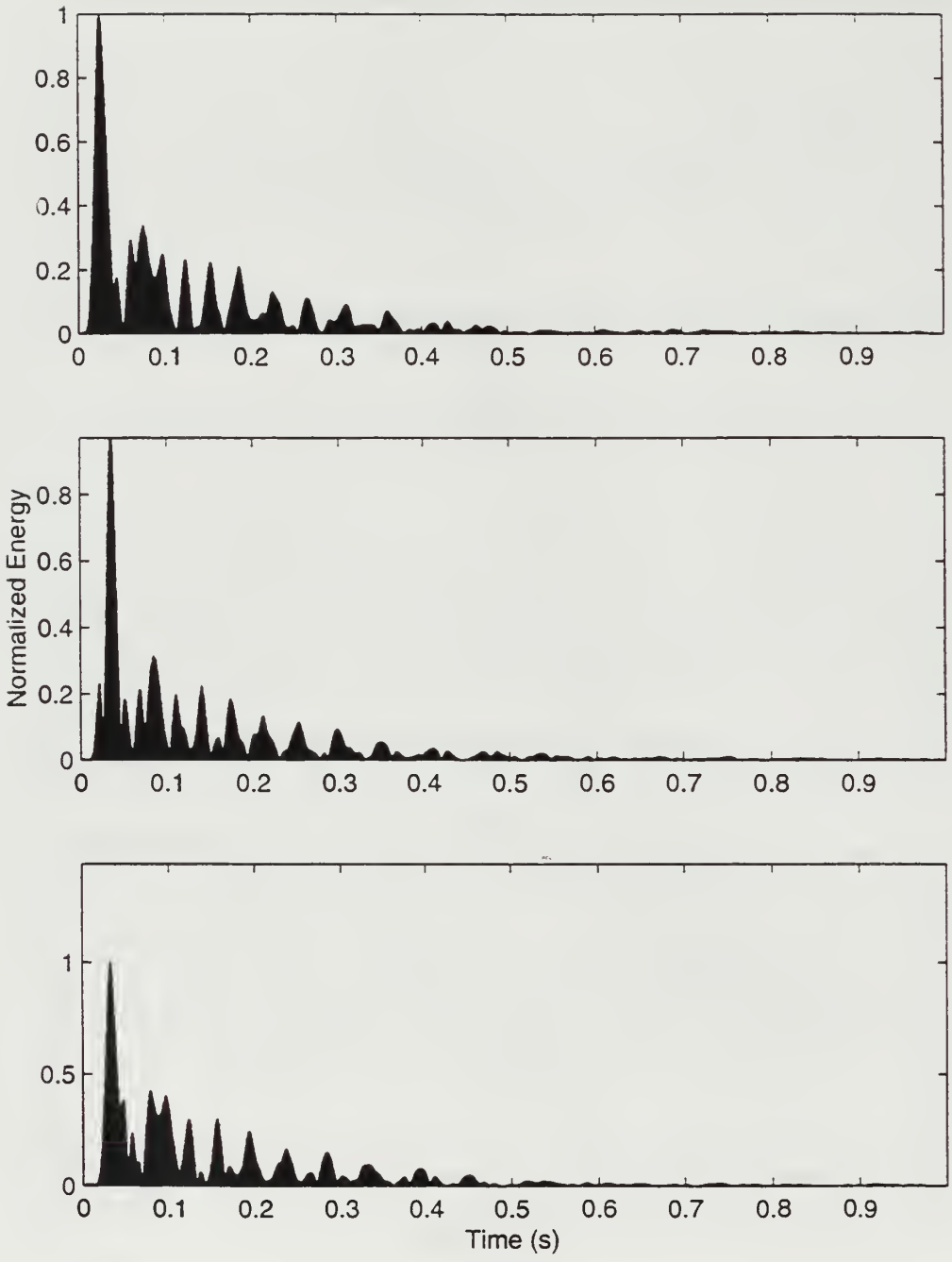

Figure 23. From top to bottom, the graphs depict time domain information of 14, 16, and 18 km output pulses for run 7.

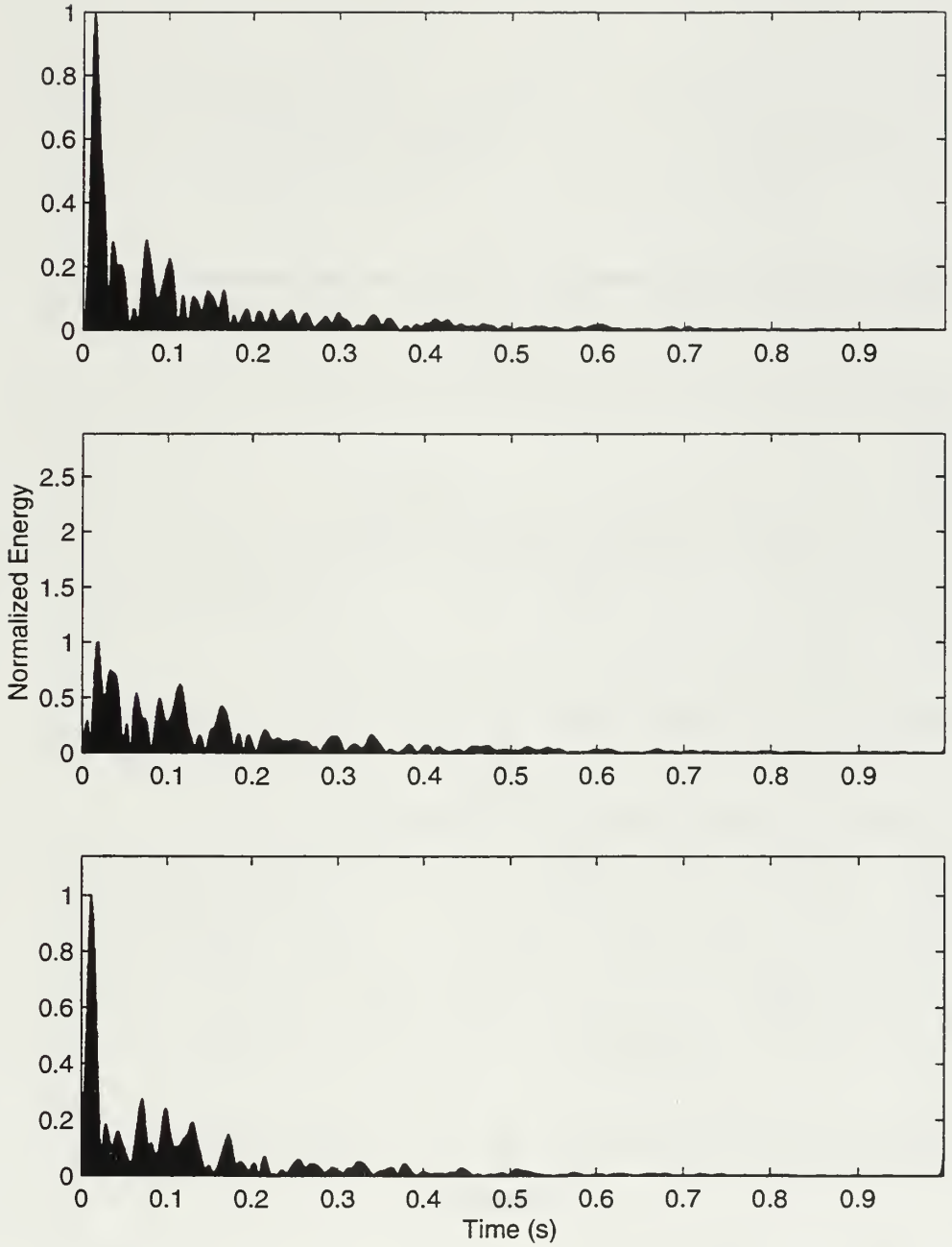

Figure 24. From top to bottom, the graphs depict time domain information of 14, 16, and 18 km output pulses for run 8.

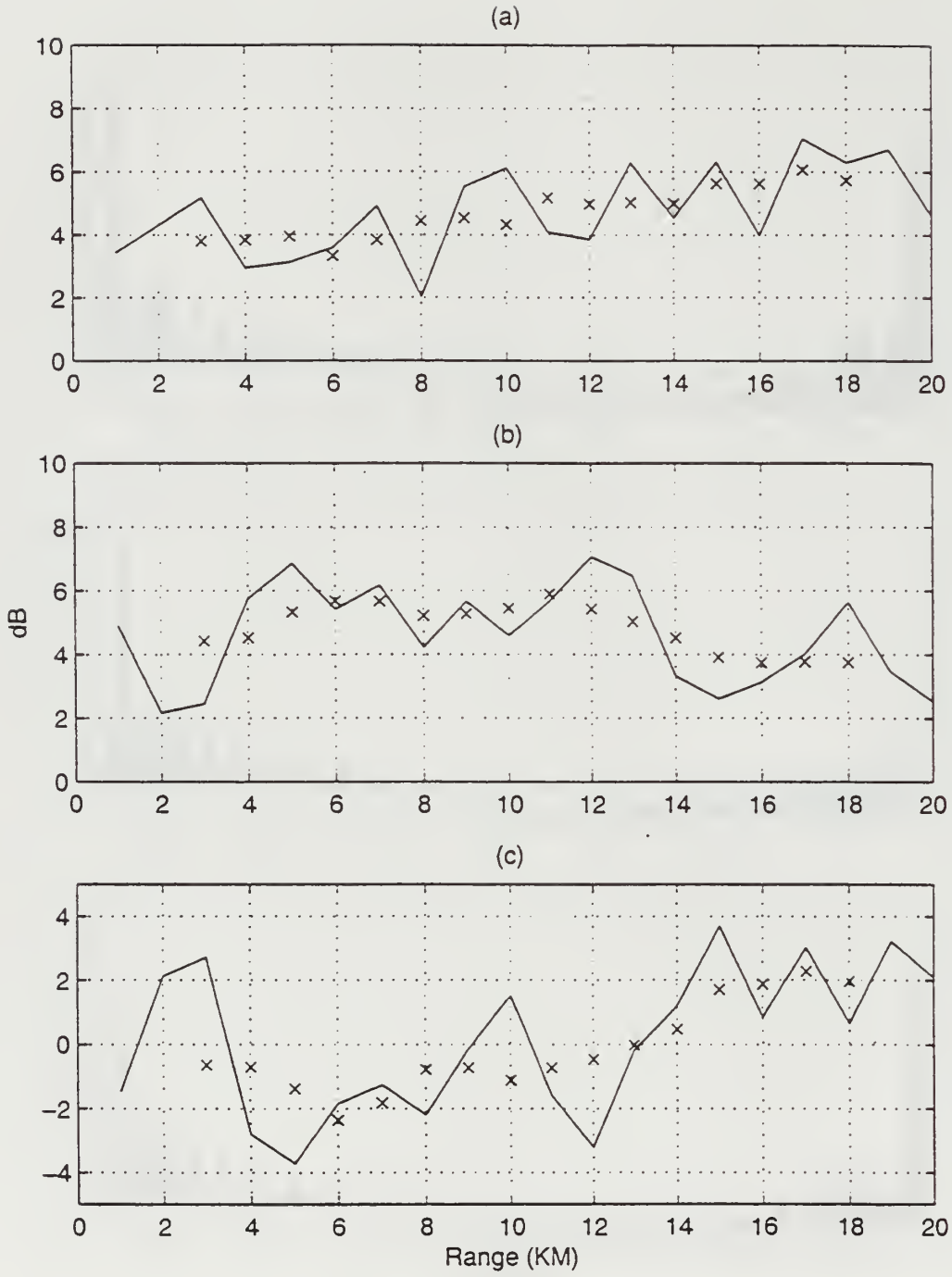

Figure 25. (a) ESL plotted for run 3. (b) ESL plotted for run 4. (c) The ESL difference between run 3 and run 4. (Graph properties as before)

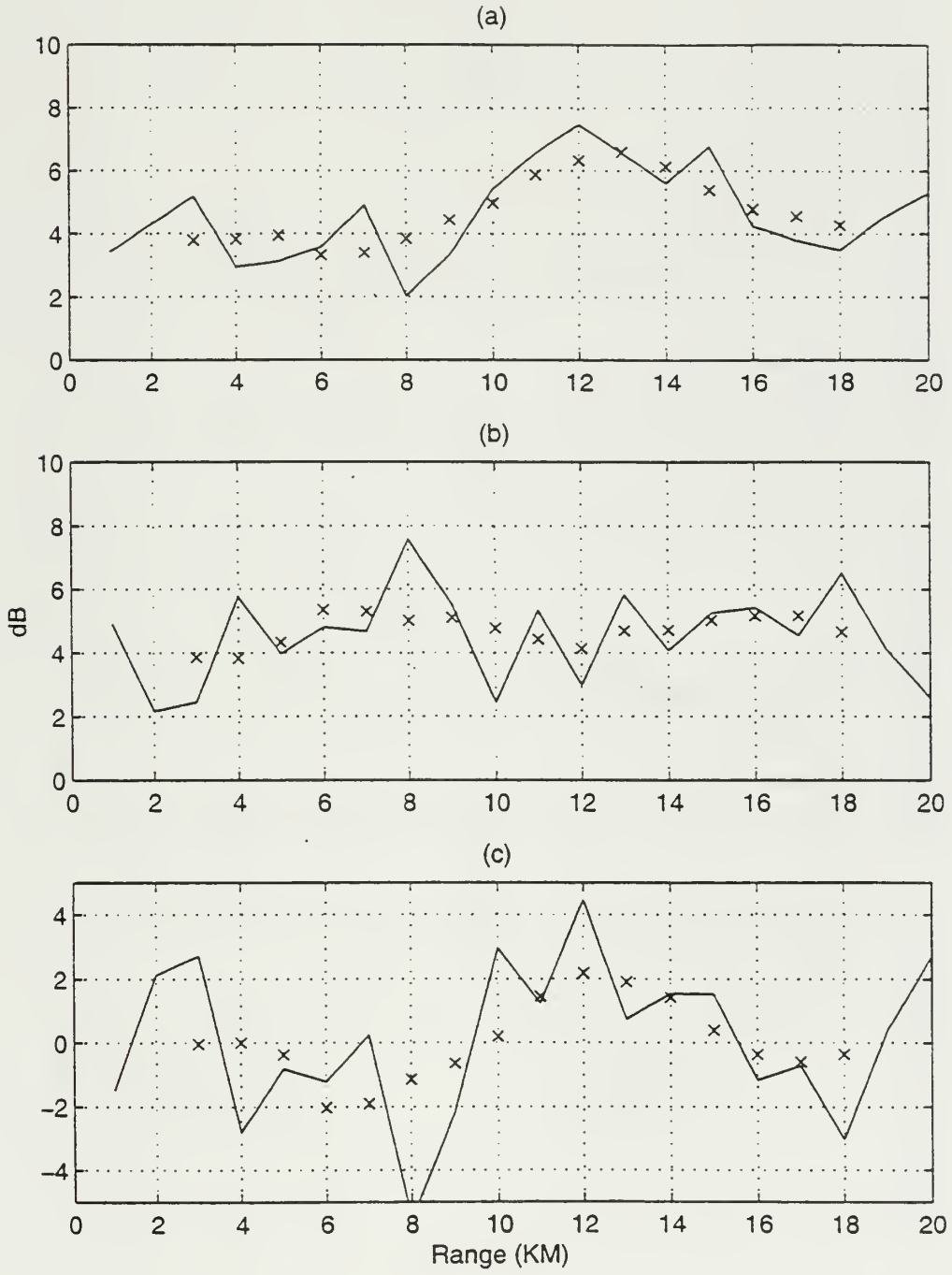

Figure 26. (a) ESL plotted for run 1. (b) ESL plotted for run 2. (c) The ESL difference between run 1 and run 2. (Graph properties as before)

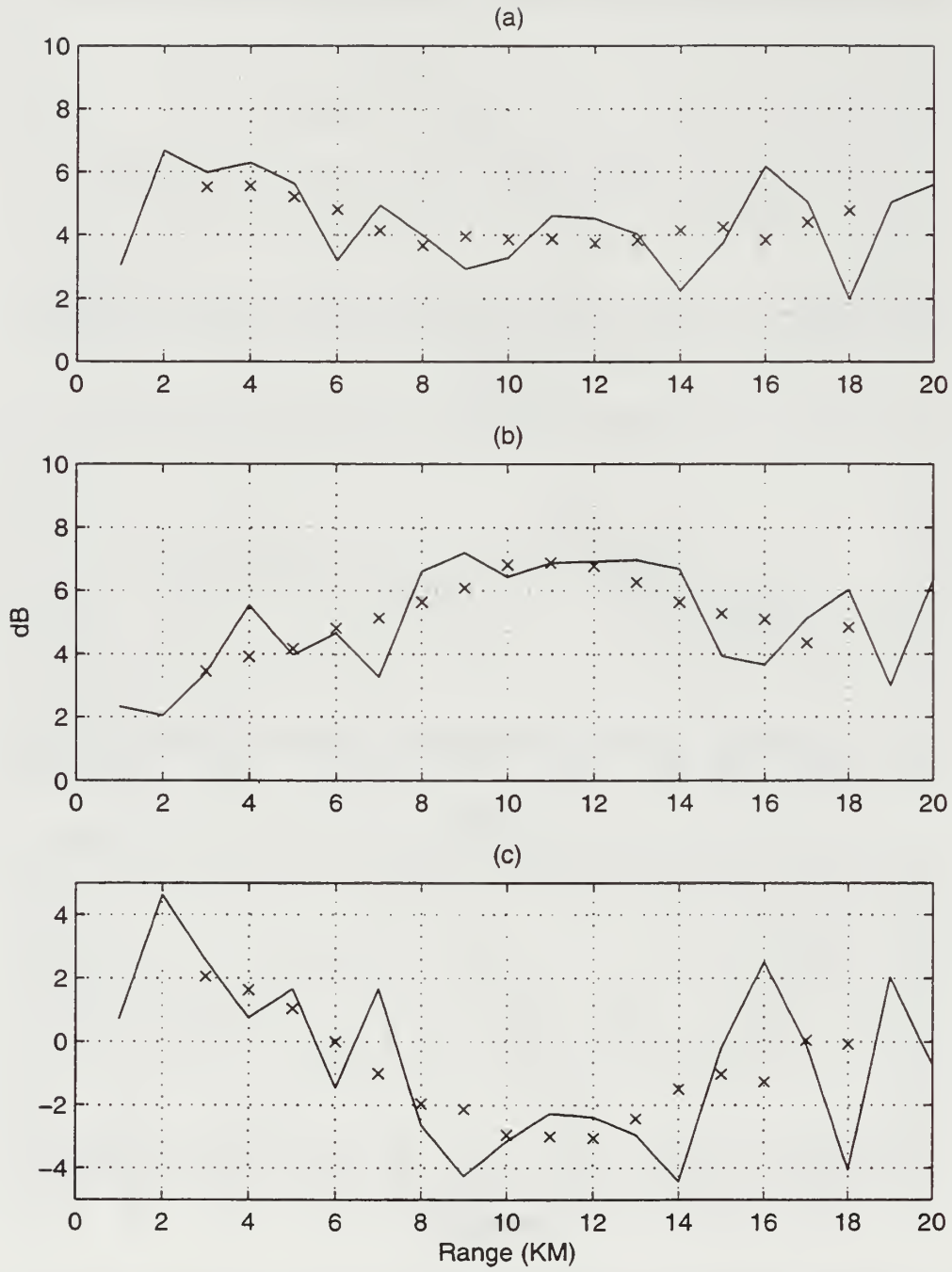

Figure 27. (a) ESL plotted for run 8. (b) ESL plotted for run 9. (c) The ESL difference between run 8 and run 9. (Graph properties as before)

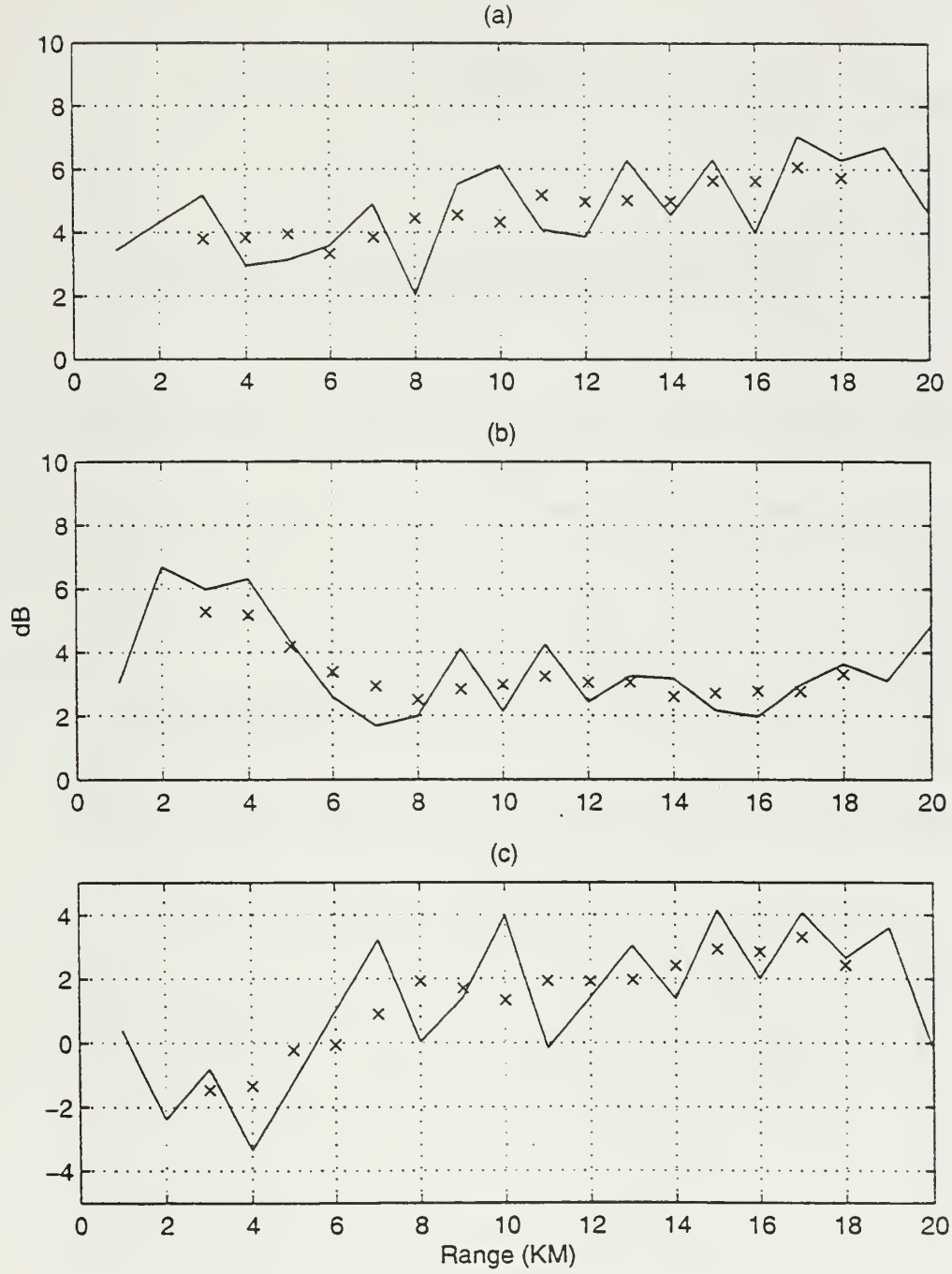

Figure 28. (a) ESL plotted for run 3. (b) ESL plotted for run 7. (c) The ESL difference between run 3 and run 7. (Graph properties as before)

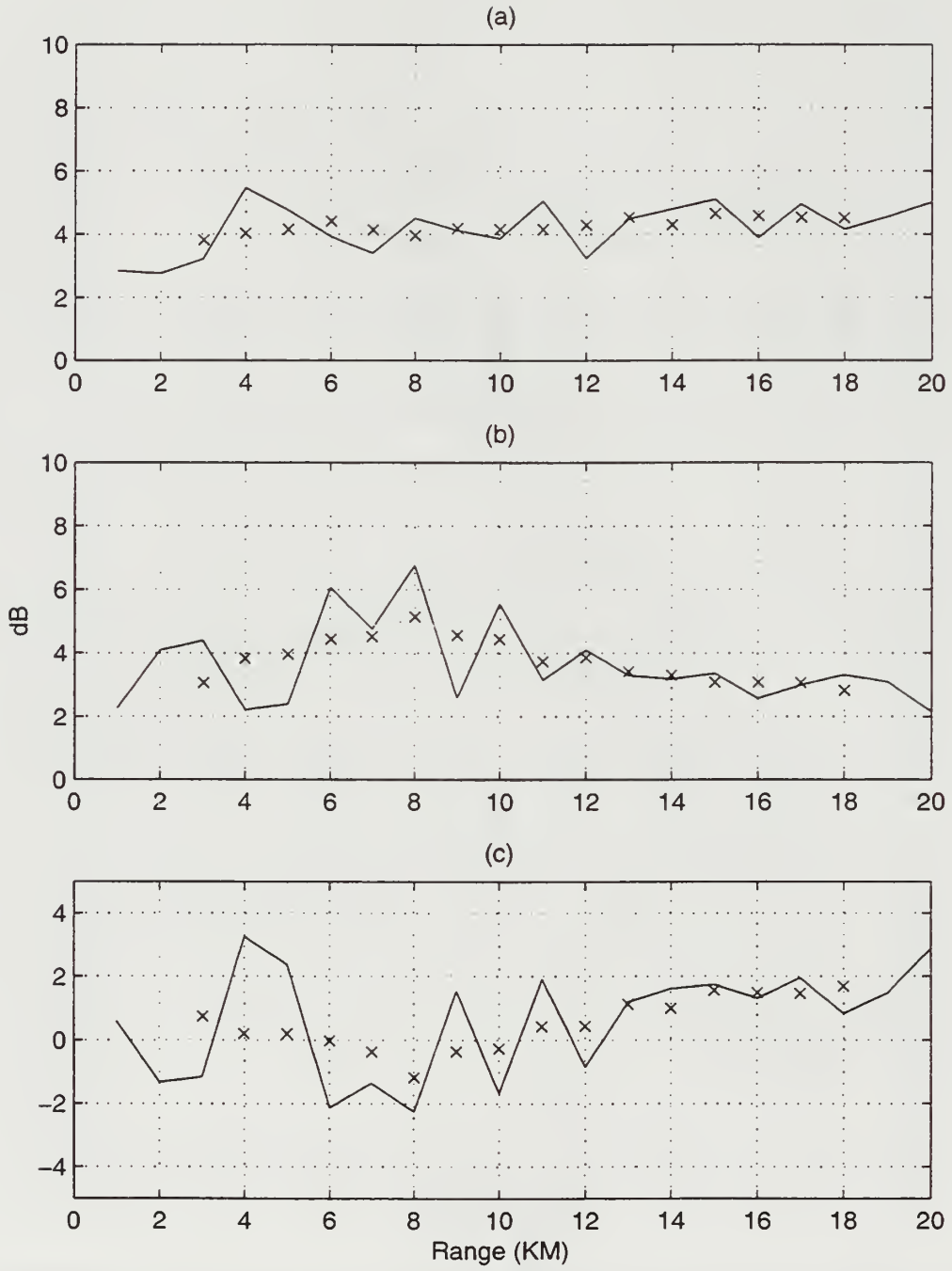

Figure 29. (a) ESL plotted for run 23. (b) ESL plotted for run 6. (c) The ESL difference between run 23 and run 6. (Graph properties as before)

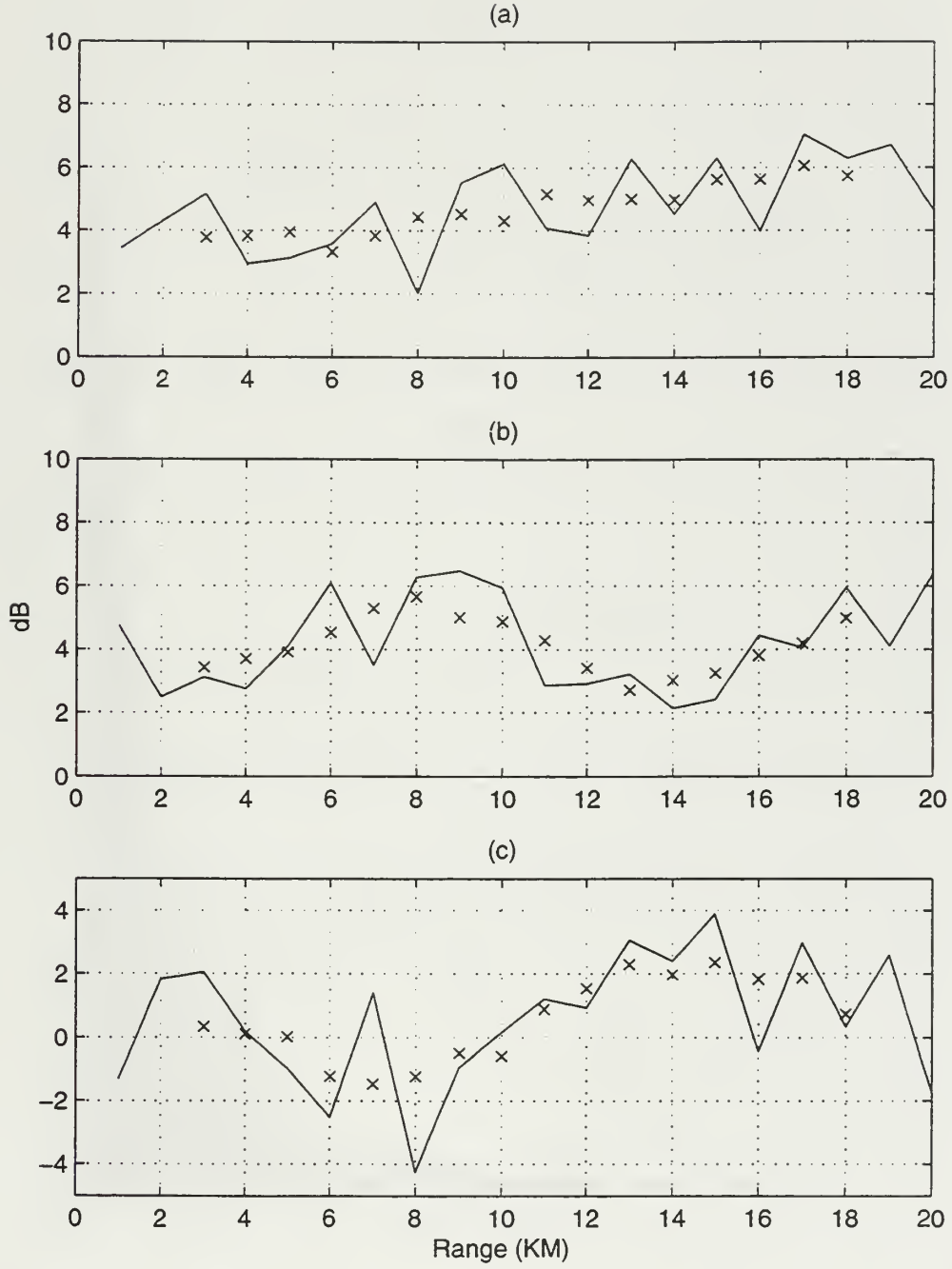

Figure 30. (a) ESL plotted for run 3 with a target depth at 11 m. (b) ESL plotted for run 3 with a target depth at 22 m. (c) The ESL difference between targets at 11 and 22 m. (Graph properties as before)

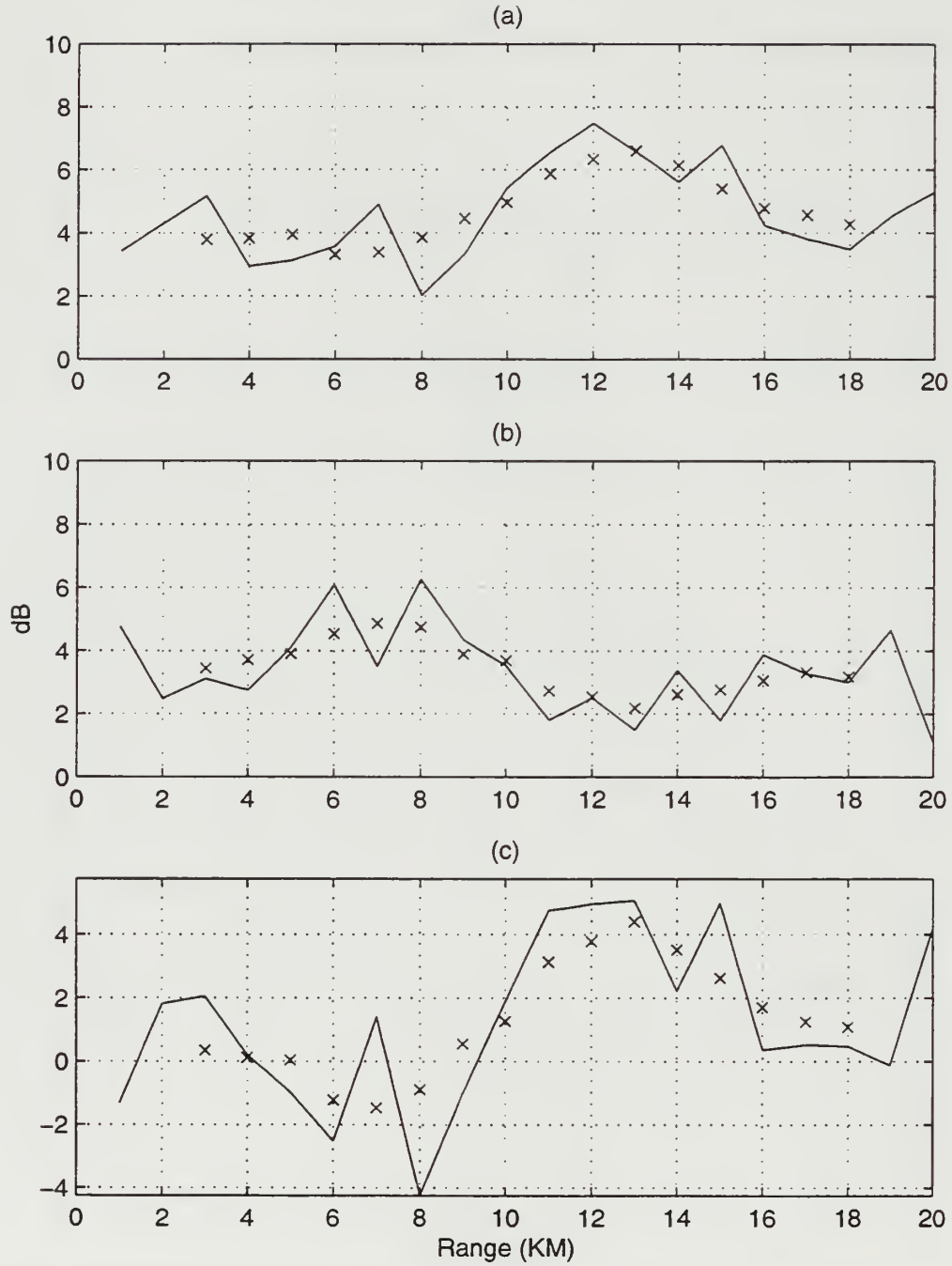

Figure 31. (a) ESL plotted for run 1 with a target depth at 11 m. (b) ESL plotted for run 1 with a target depth at 22 m. (c) The ESL difference between targets at 11 and 22 m. (Graph properties as before)

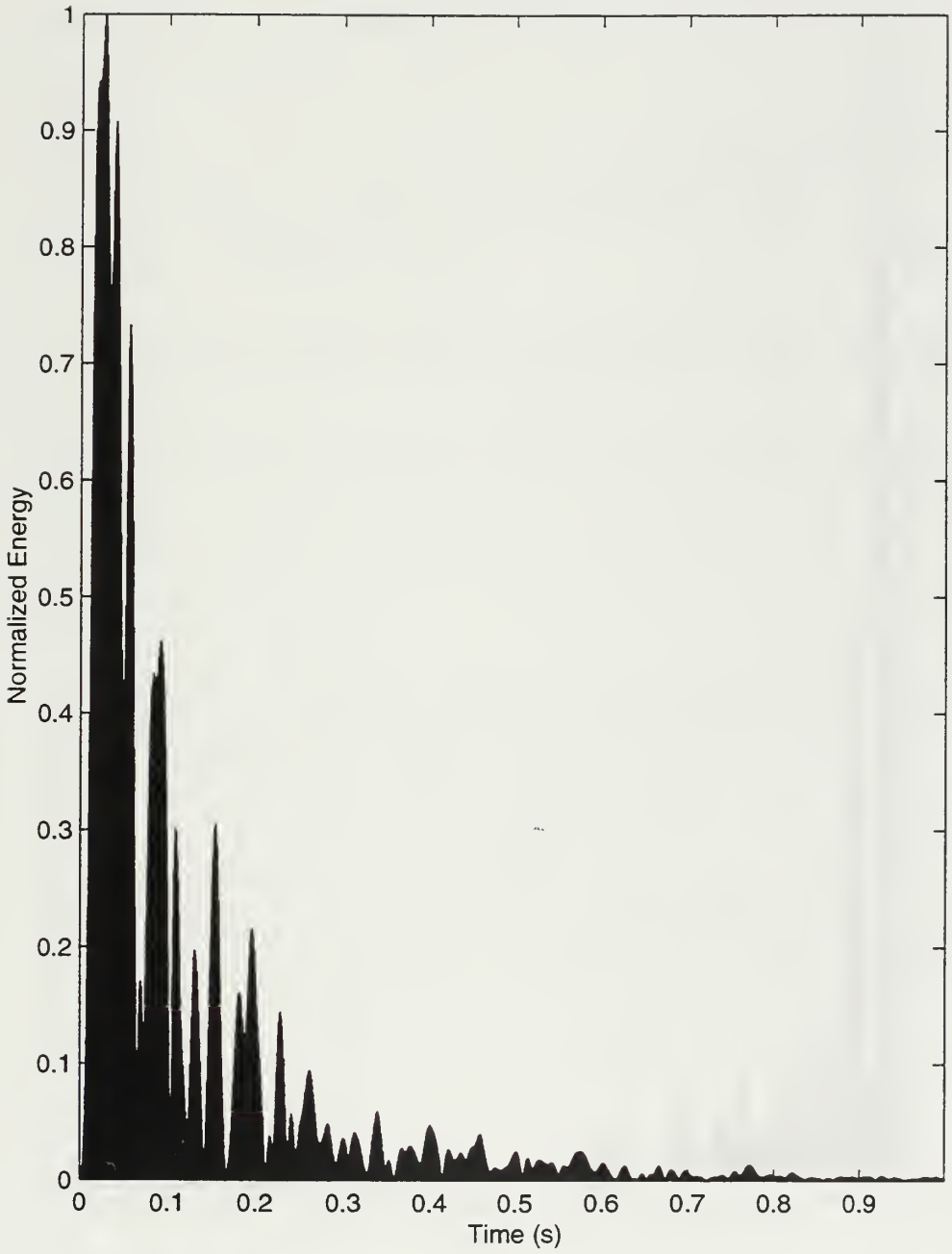

Figure 32. The graph shows the time domain information for run 1 at 13 km for a target depth of 11 m.

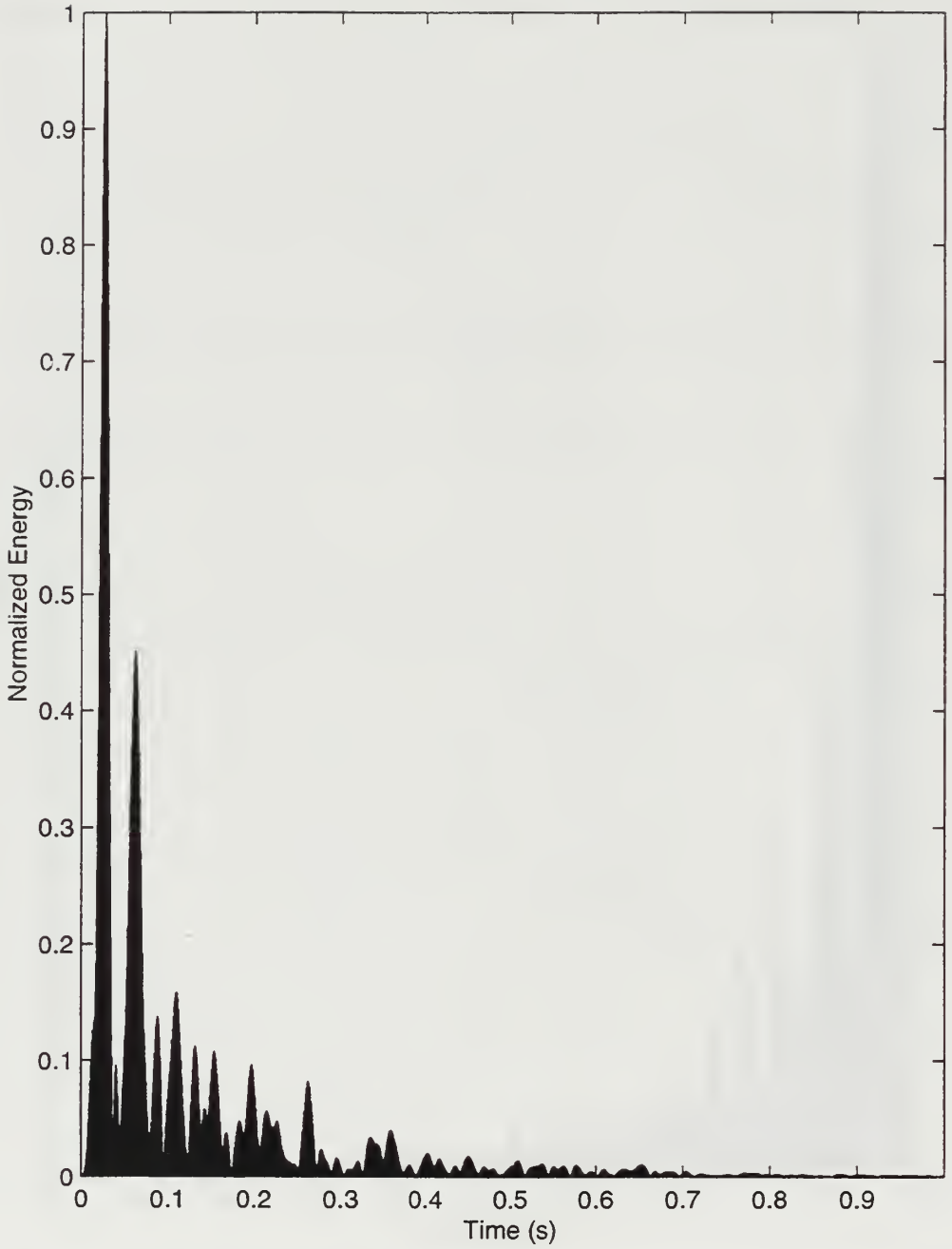

Figure 33. The graph shows the time domain information for run 1 at 13 km for a target depth of 22 m.

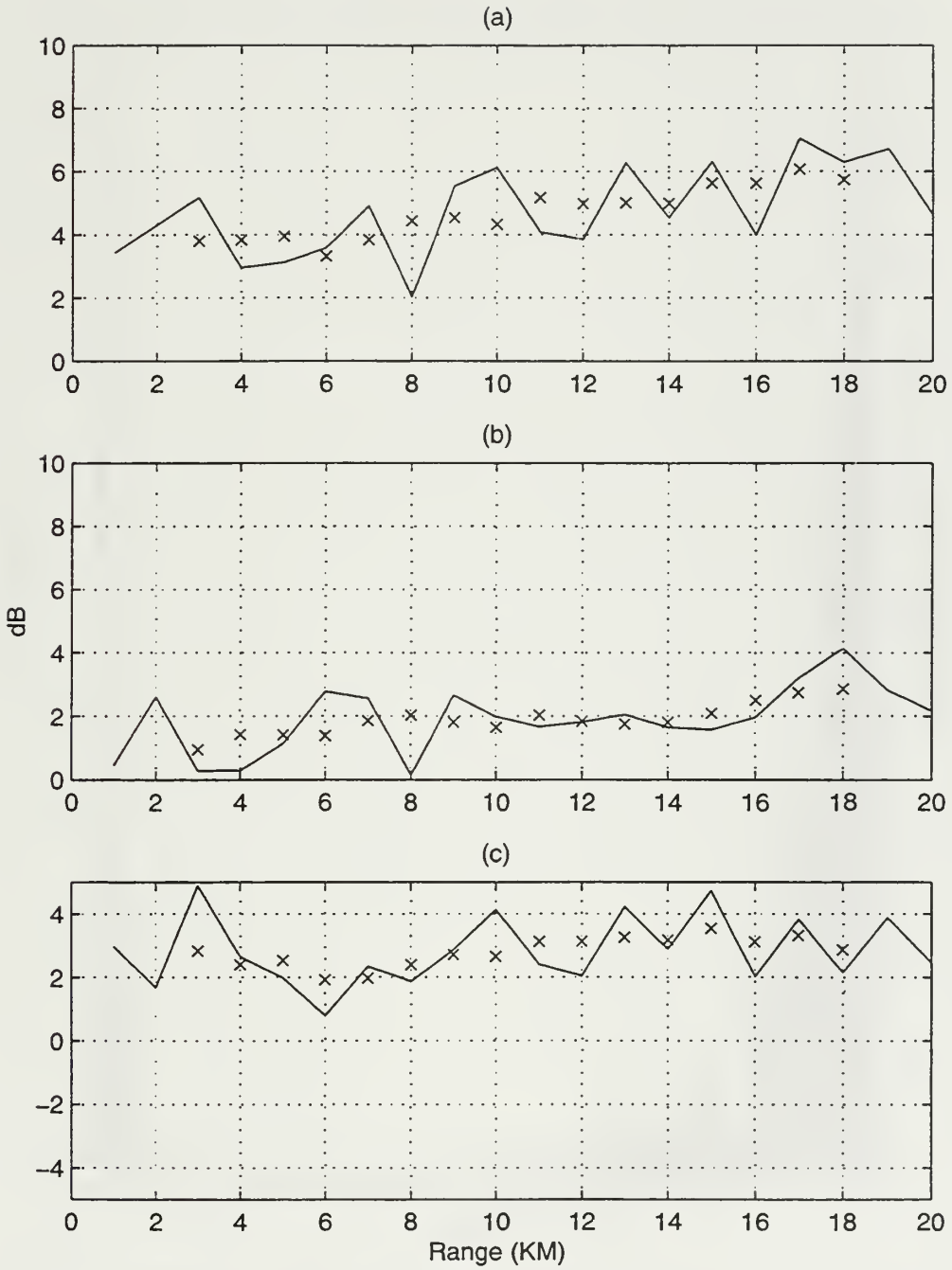

Figure 34. (a) ESL plotted for run 3. (b) ESL plotted for run 10. (c) The ESL difference between run 3 and run 10. (Graph properties as before)

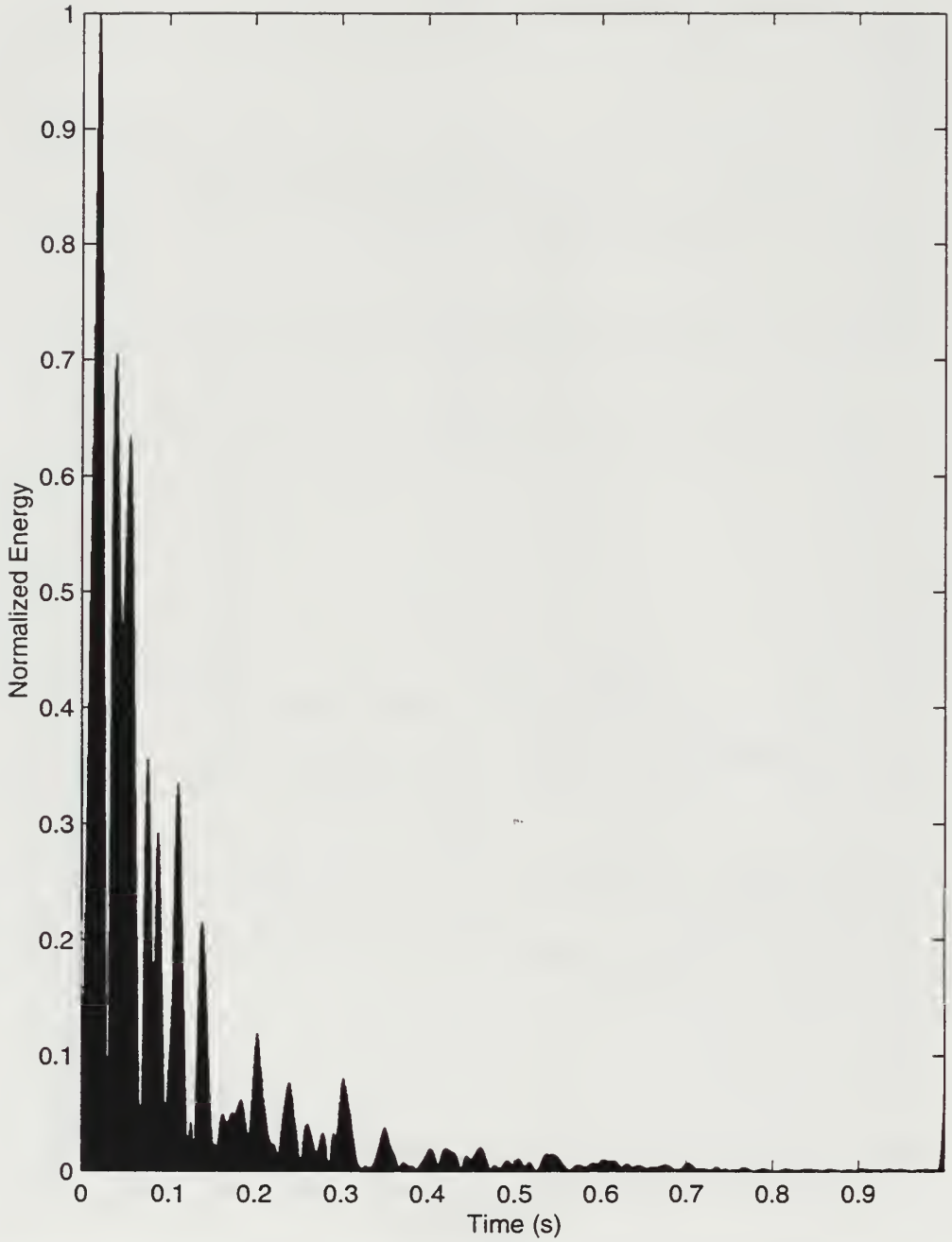

Figure 35. The graph shows the time domain information for run 3 at 20 km.

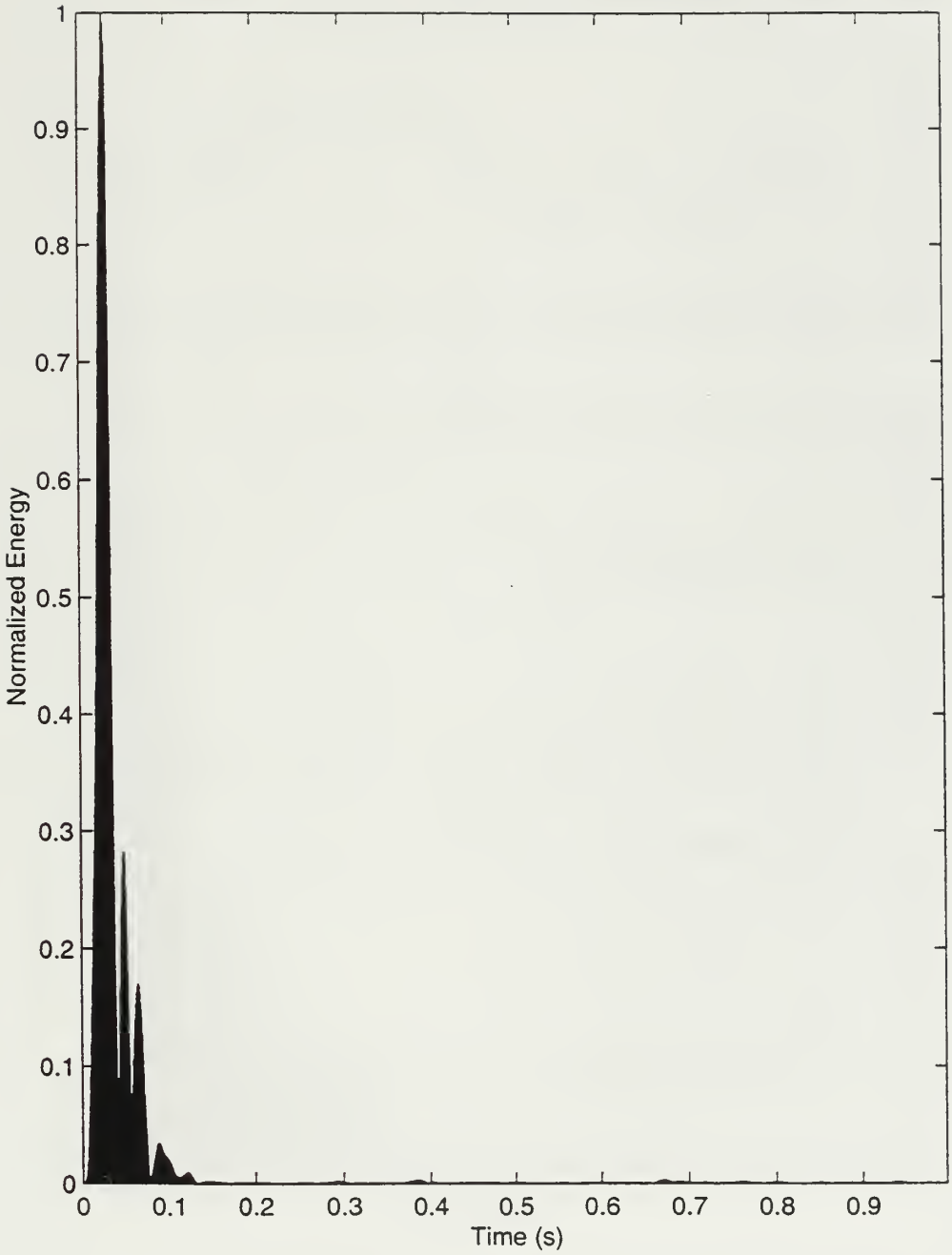

Figure 36. The graph shows the time domain information for run 10 at 20 km.

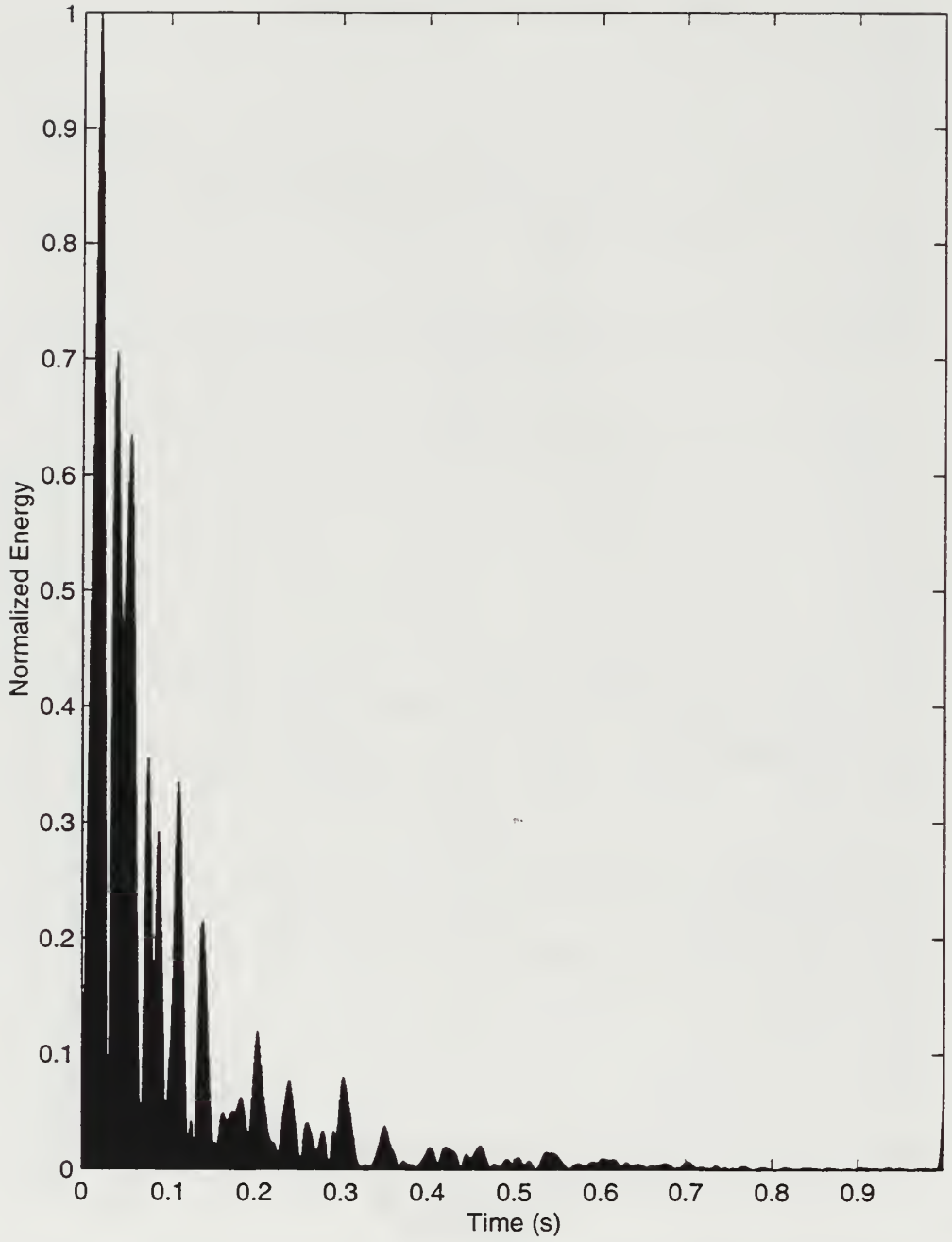

Figure 35. The graph shows the time domain information for run 3 at 20 km.

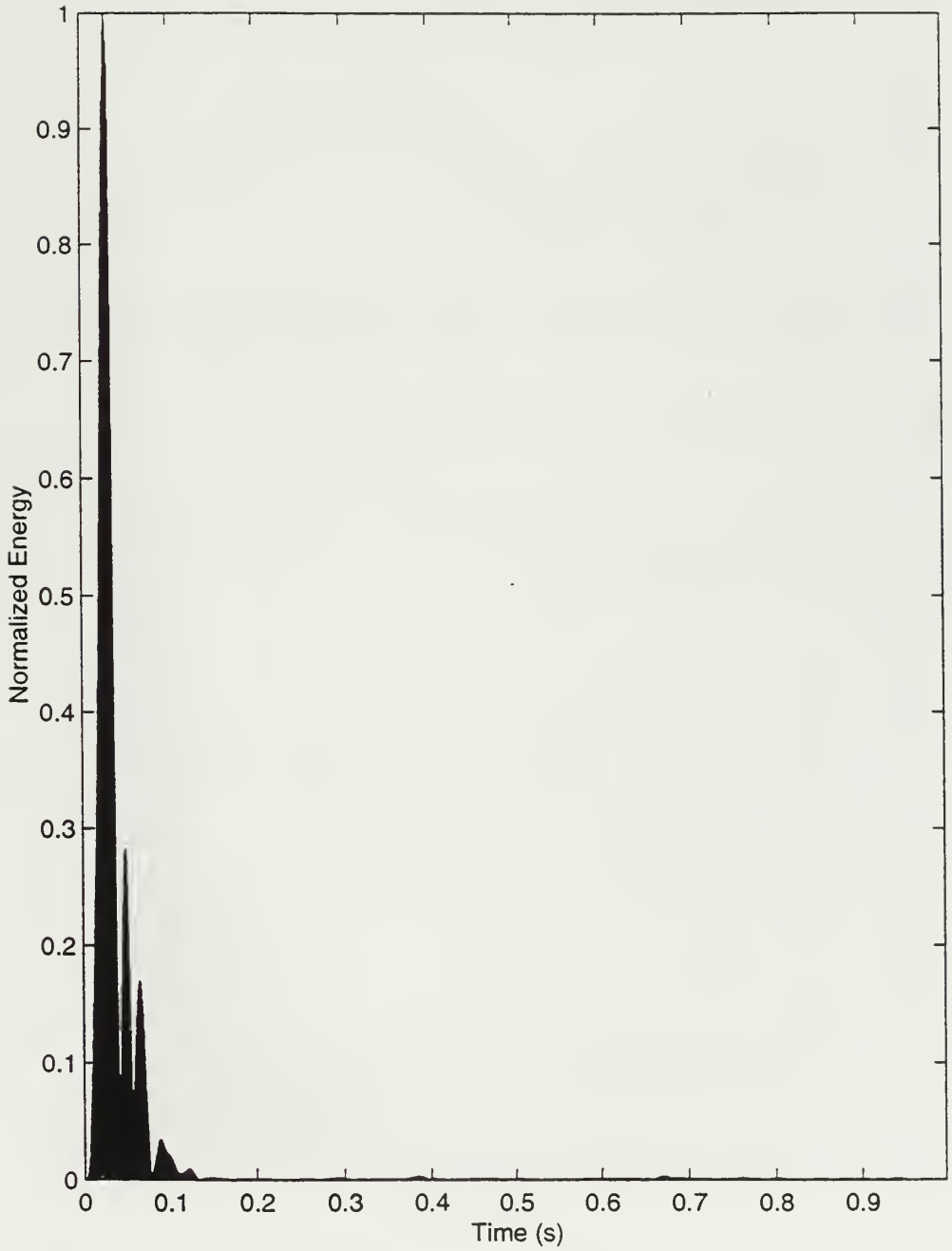

Figure 36. The graph shows the time domain information for run 10 at 20 km.

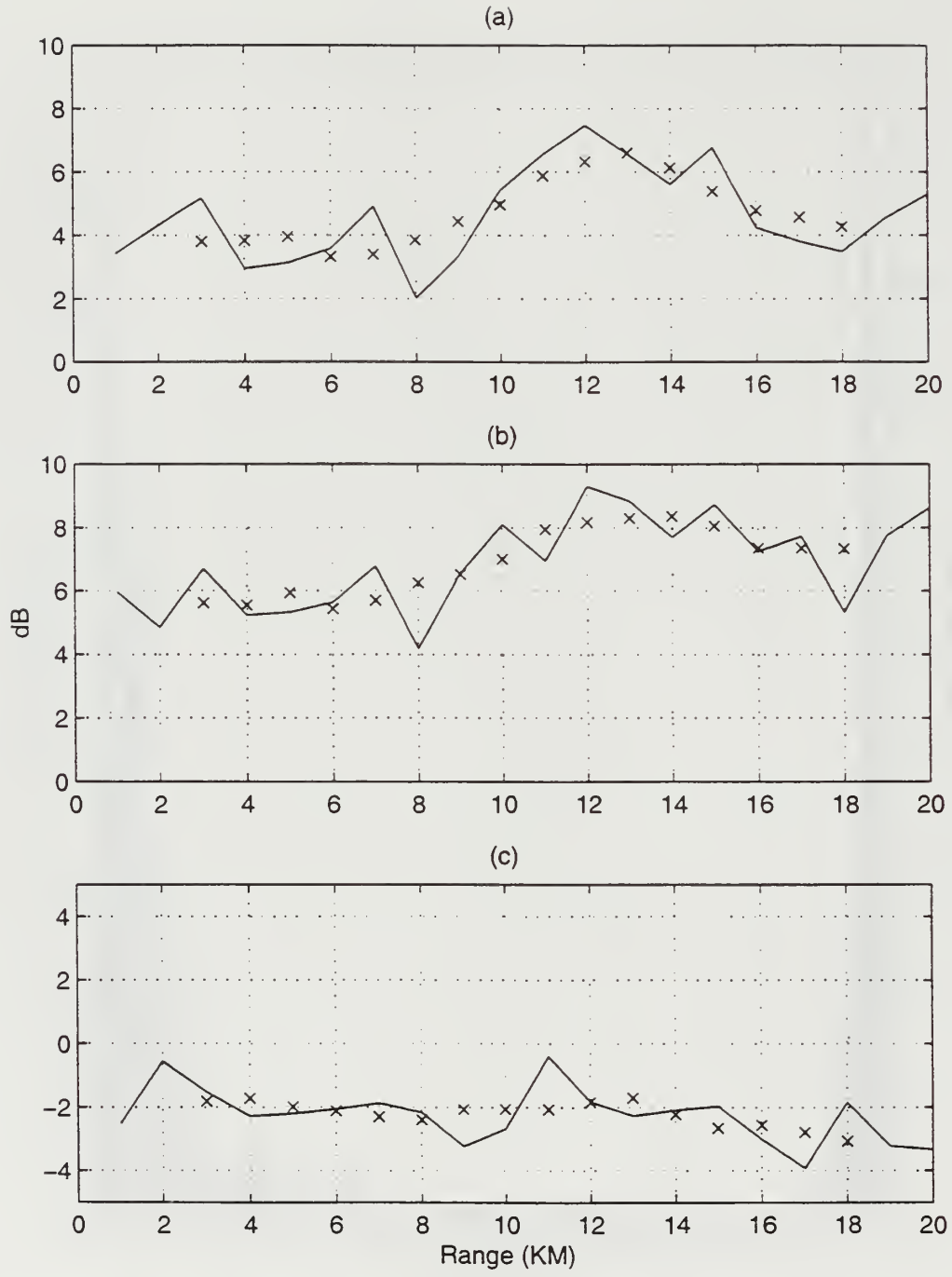

Figure 37. (a) ESL plotted for run 1. (b) ESL plotted for run 11. (c) The ESL difference between run 1 and run 11. (Graph properties as before)

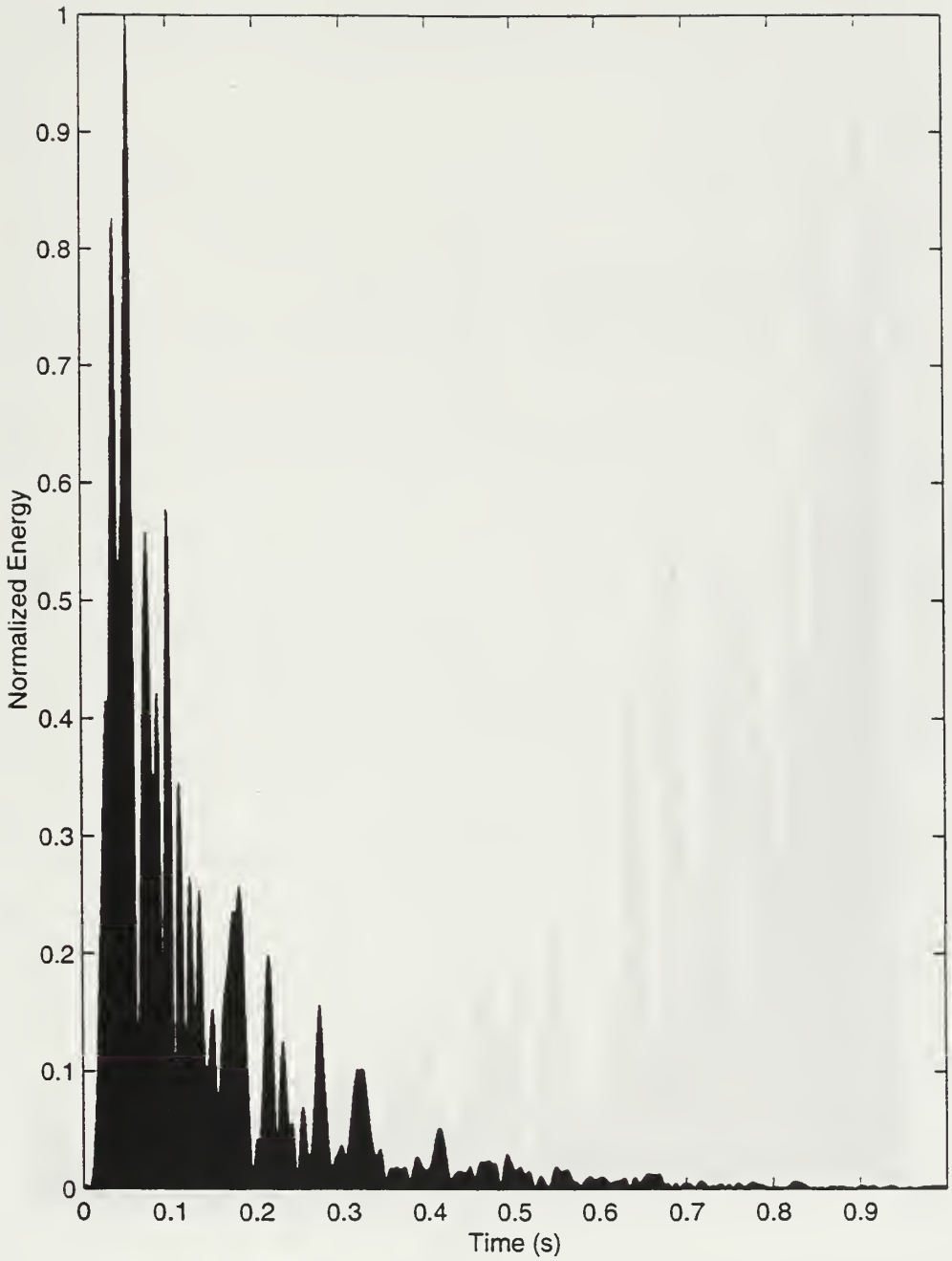

Figure 38. The graph shows the time domain information for run 1 at 20 km.

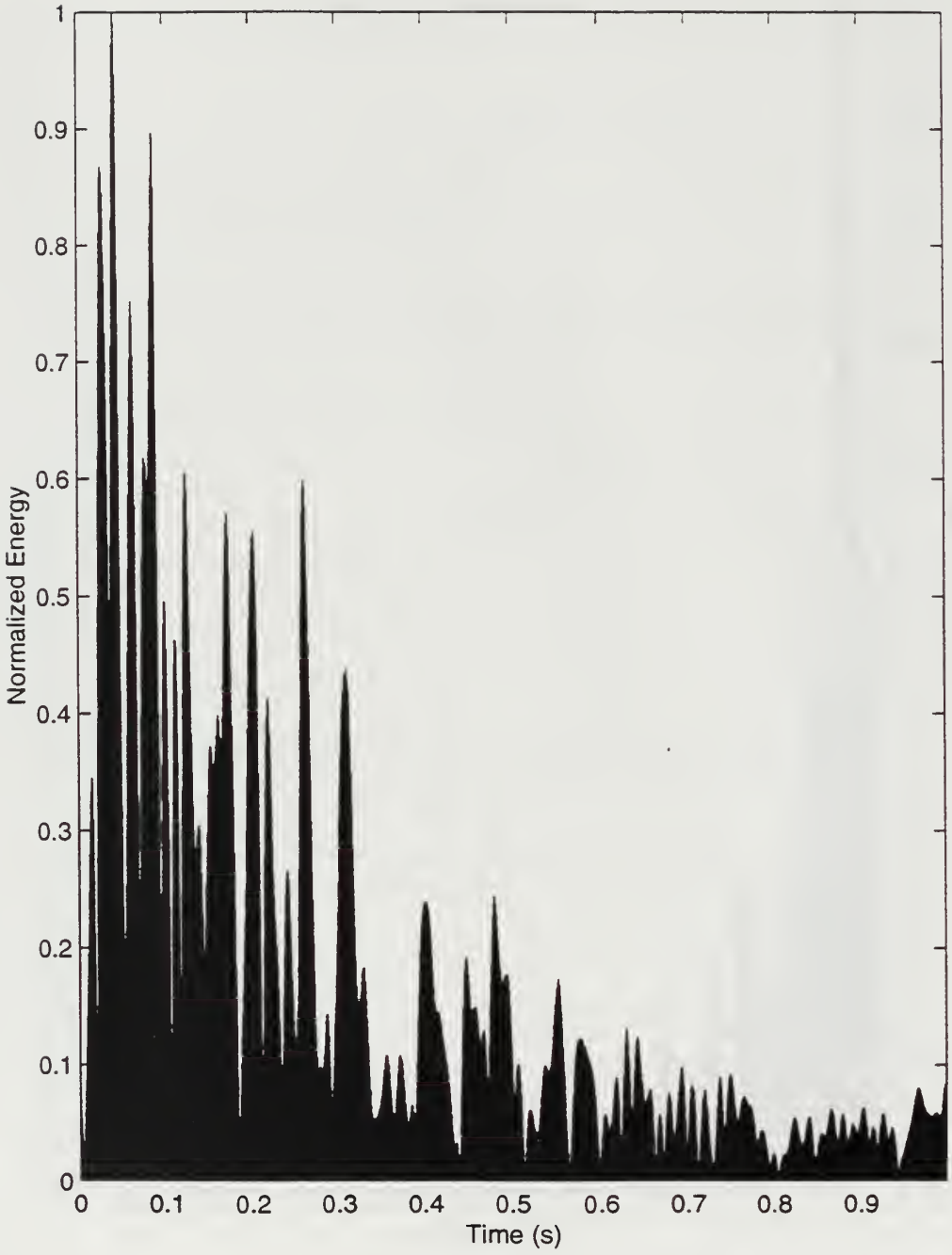

Figure 39. The graph shows the time domain information for run 11 at 20 km.

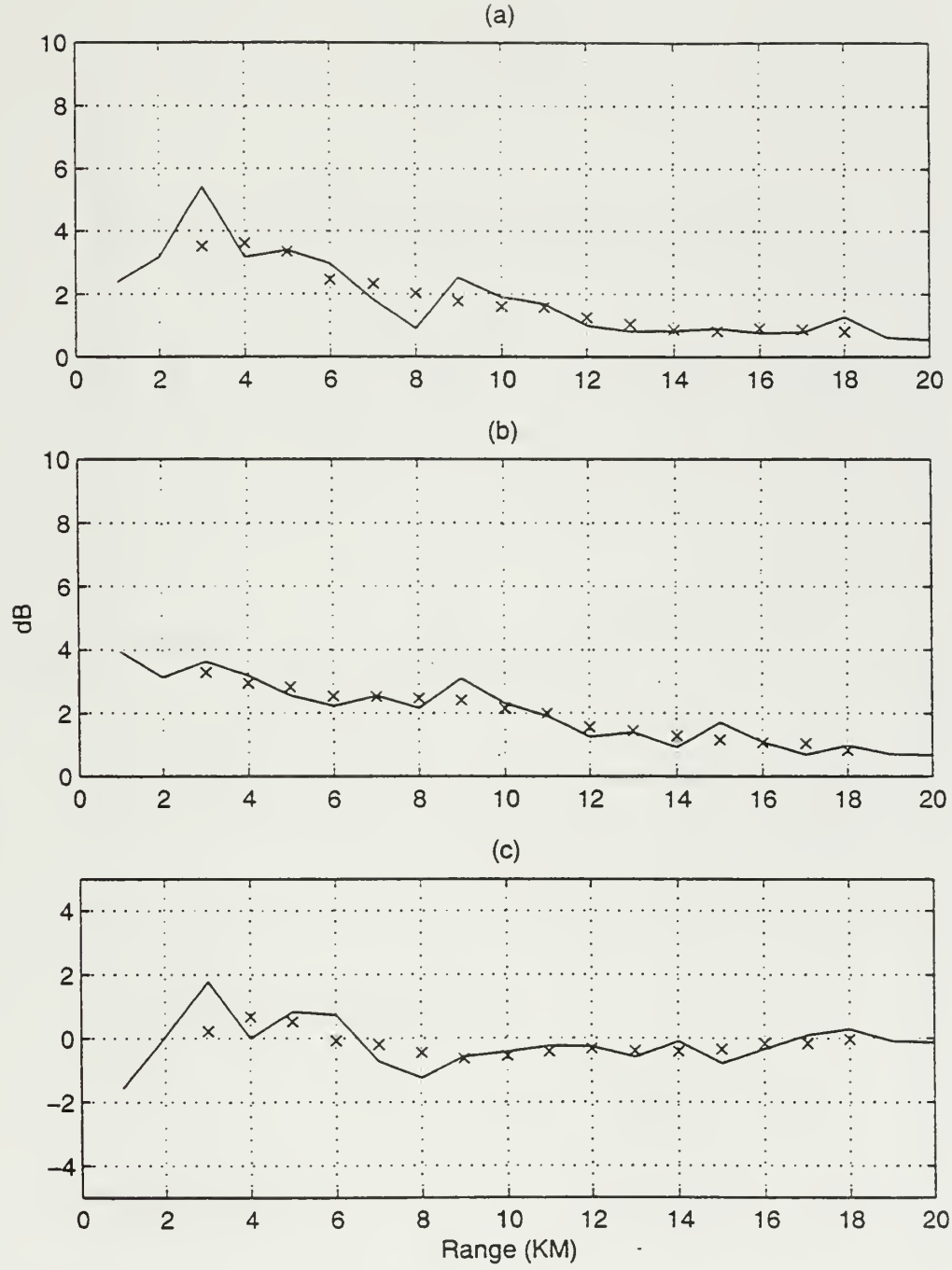

Figure 40. (a) ESL plotted for run 21. (b) ESL plotted for run 22. (c) The ESL difference between run 21 and run 22. (Graph properties as before)

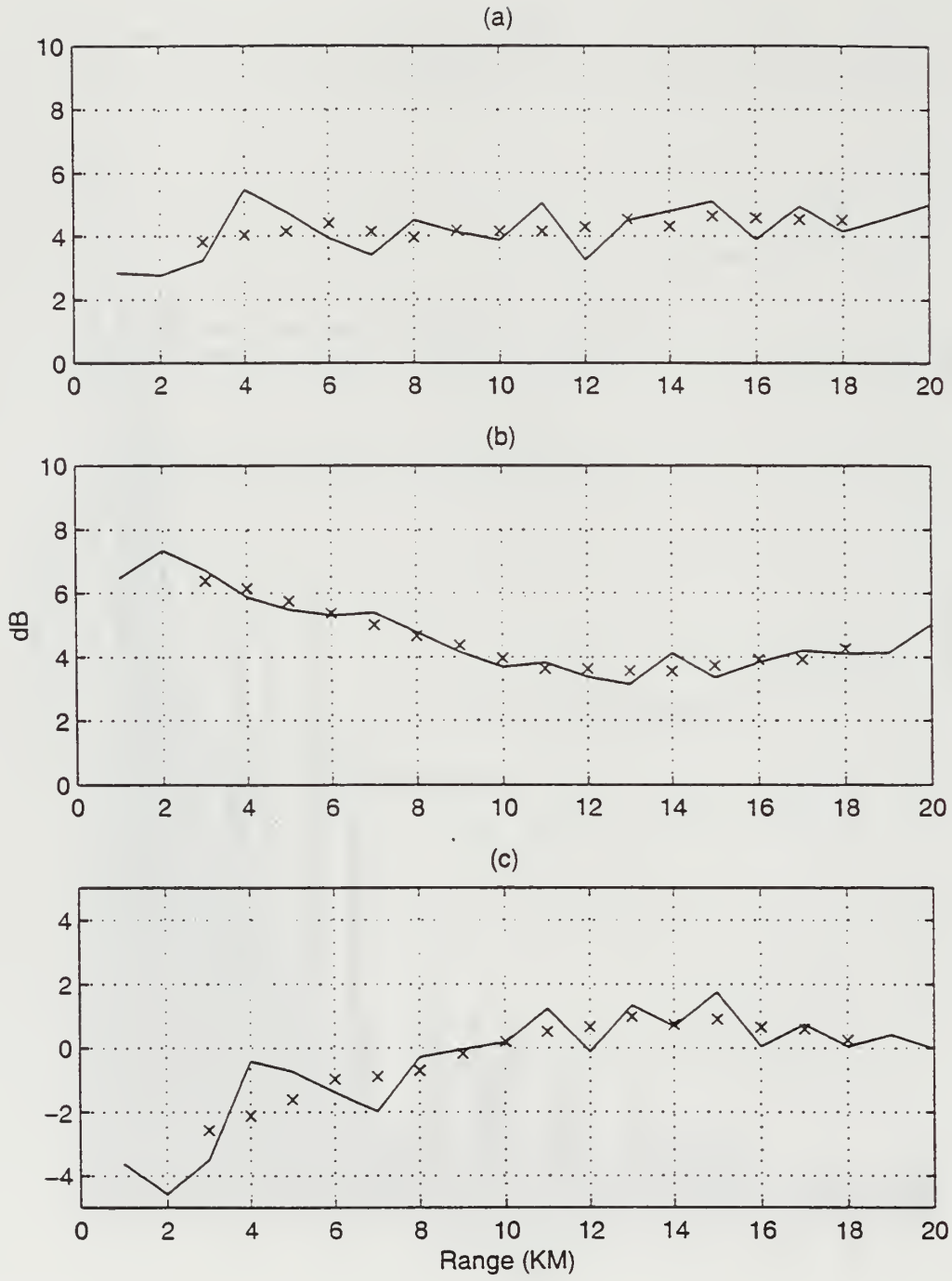

Figure 41. (a) ESL plotted for run 23. (b) ESL plotted for run 24. (c) The ESL difference between run 23 and run 24. (Graph properties as before)

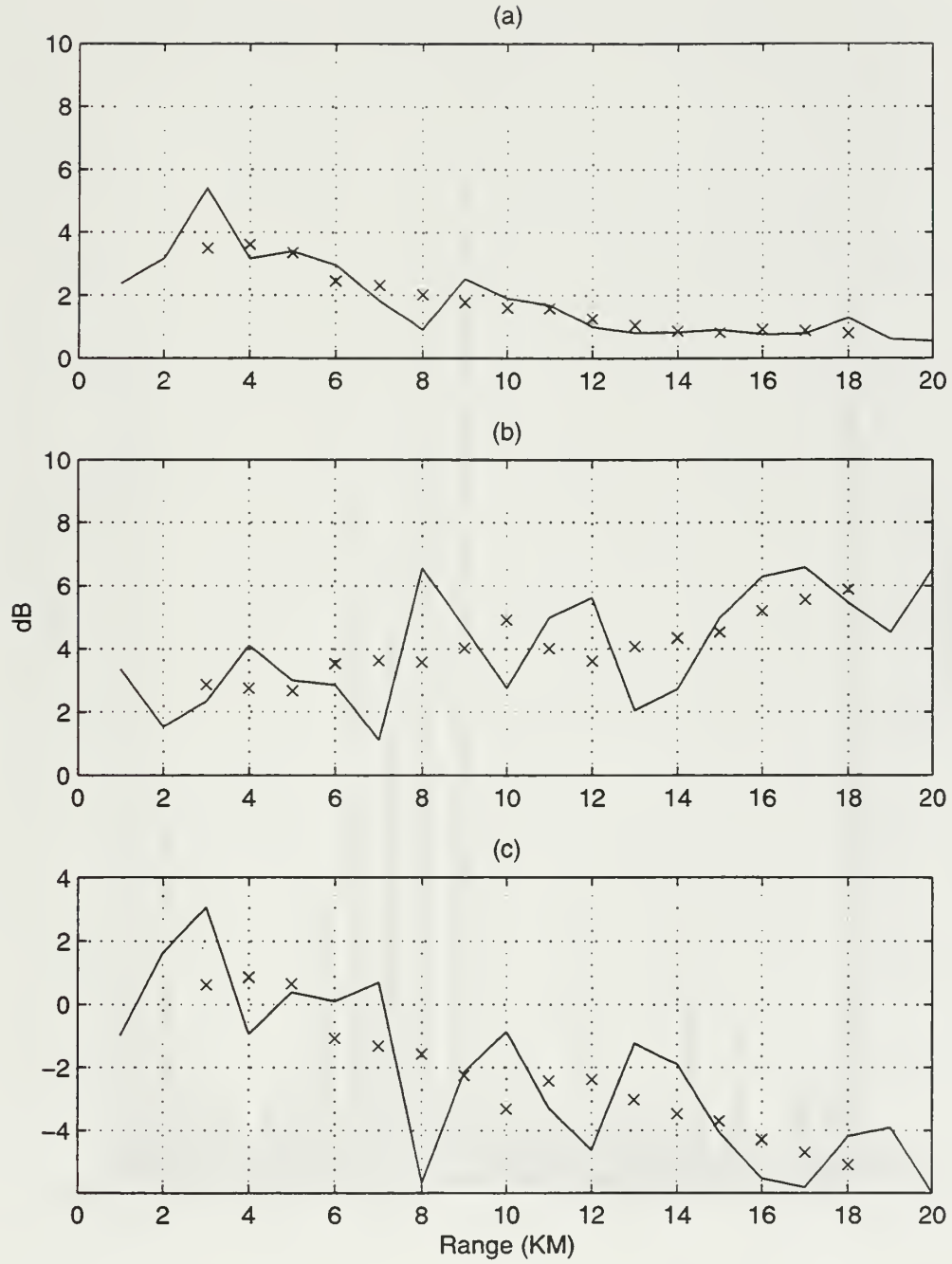

Figure 42. (a) ESL plotted for run 21 with a target depth at 11 m. (b) ESL plotted for run 21 with a target depth at 33 m. (c) The ESL difference between targets at 11 and 33 m. (Graph properties as before)

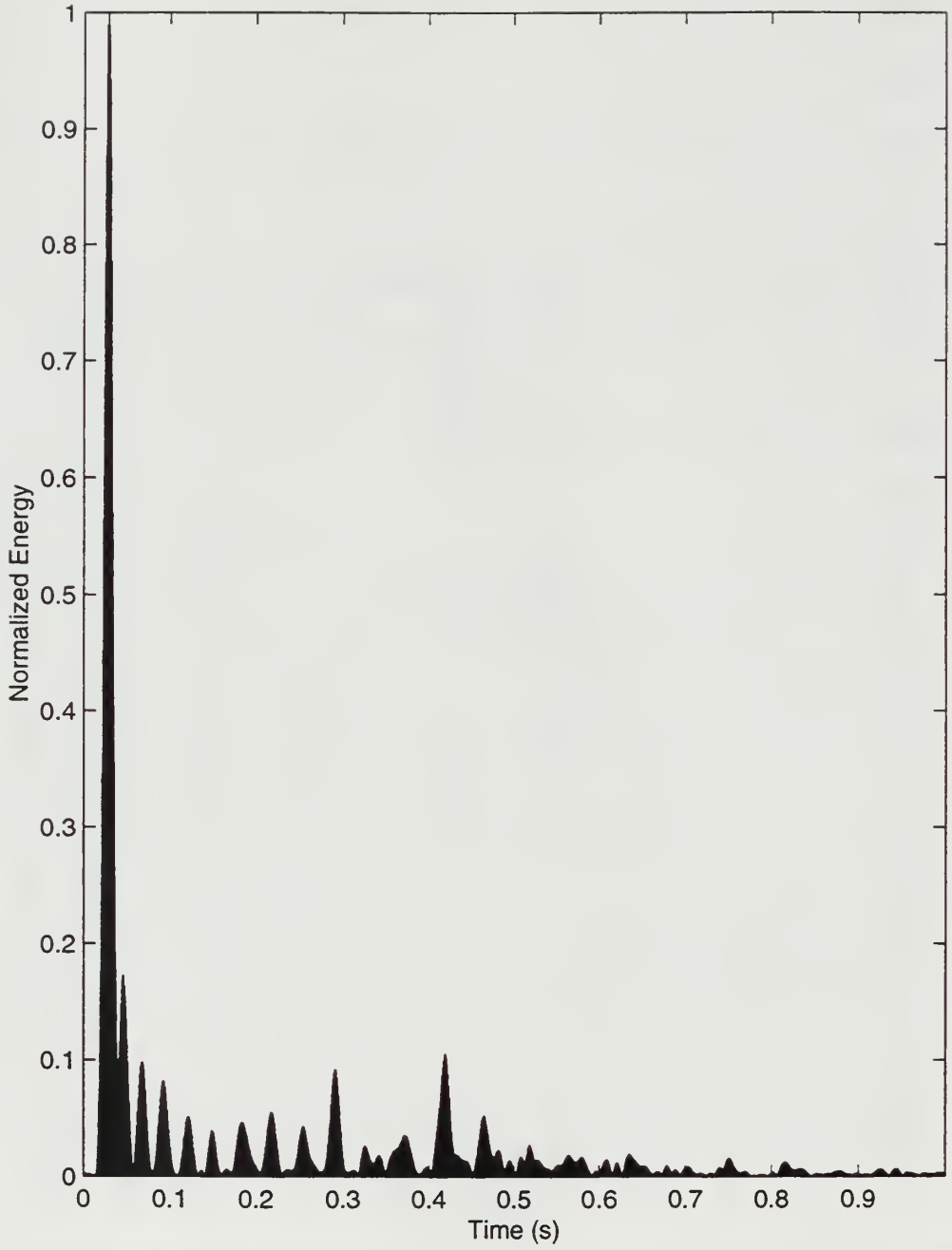

Figure 43. The graph shows the time domain information for run 21 at 20 km with a target depth of 11 m.

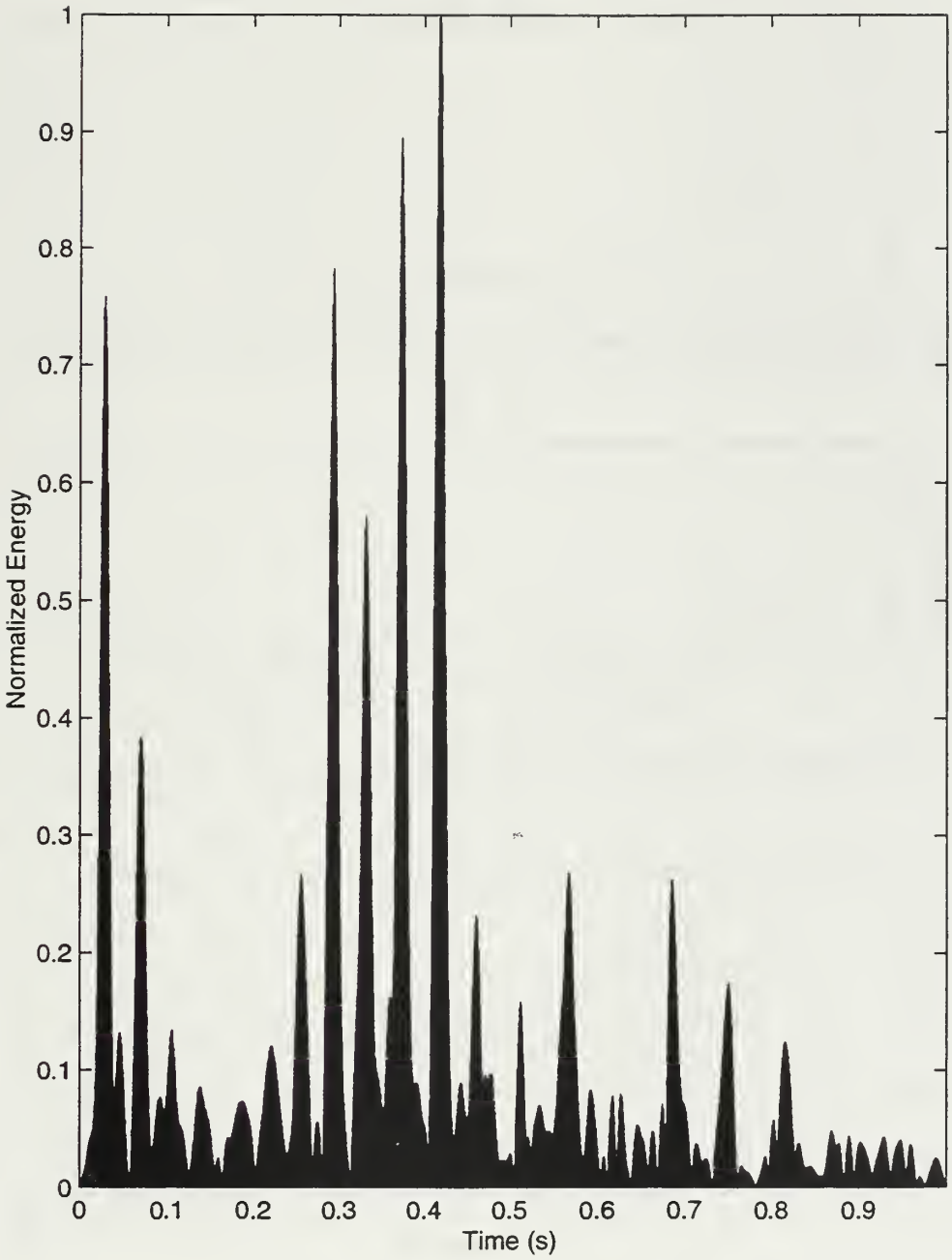

Figure 44. The graph shows the time domain information for run 21 at 20 km with a target depth of 33 m.

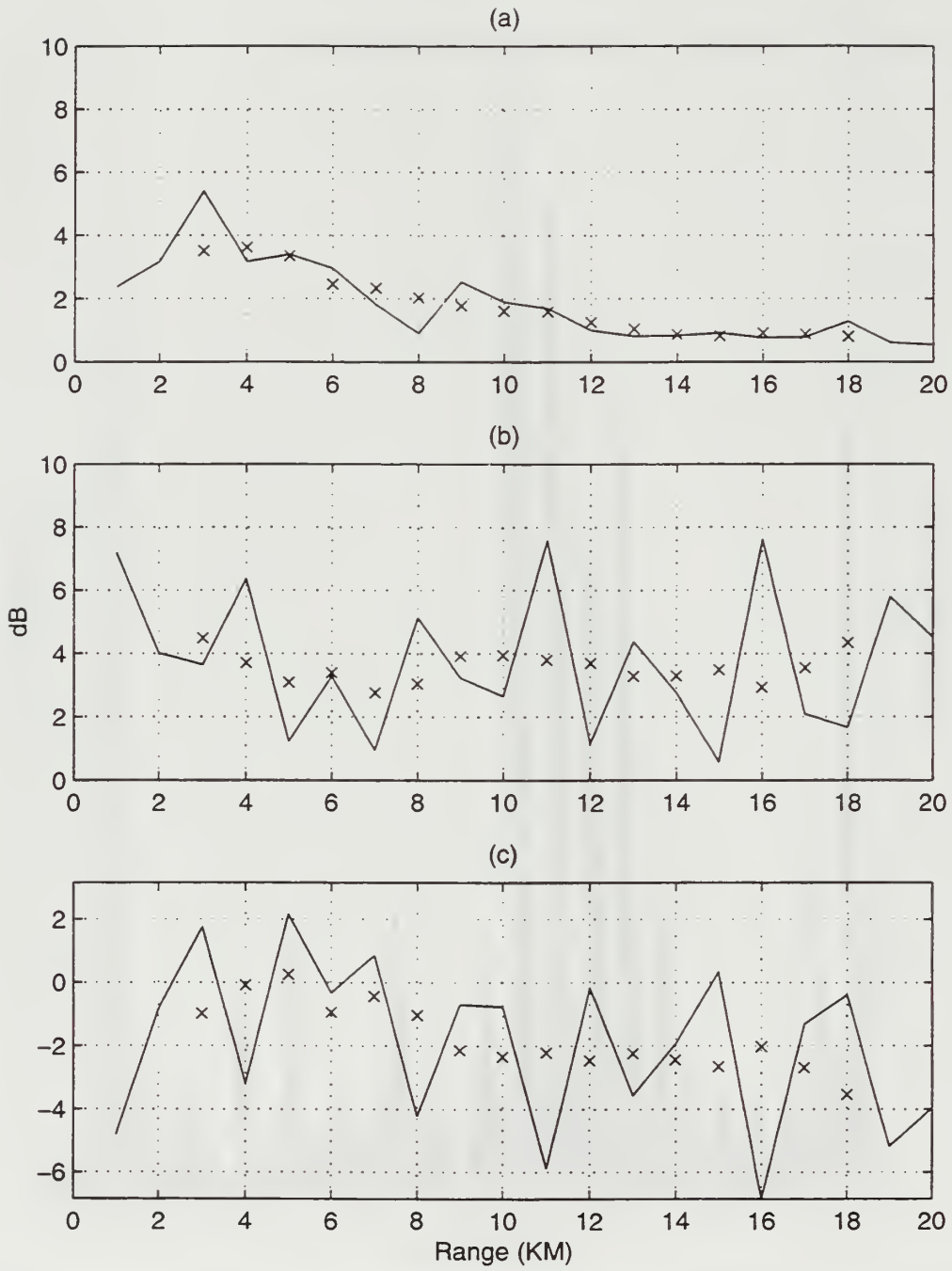

Figure 45. (a) ESL plotted for run 21 with a target depth at 11 m. (b) ESL plotted for run 21 with a target depth at 55 m. (c) The ESL difference between targets at 11 and 55 m. (Graph properties as before)

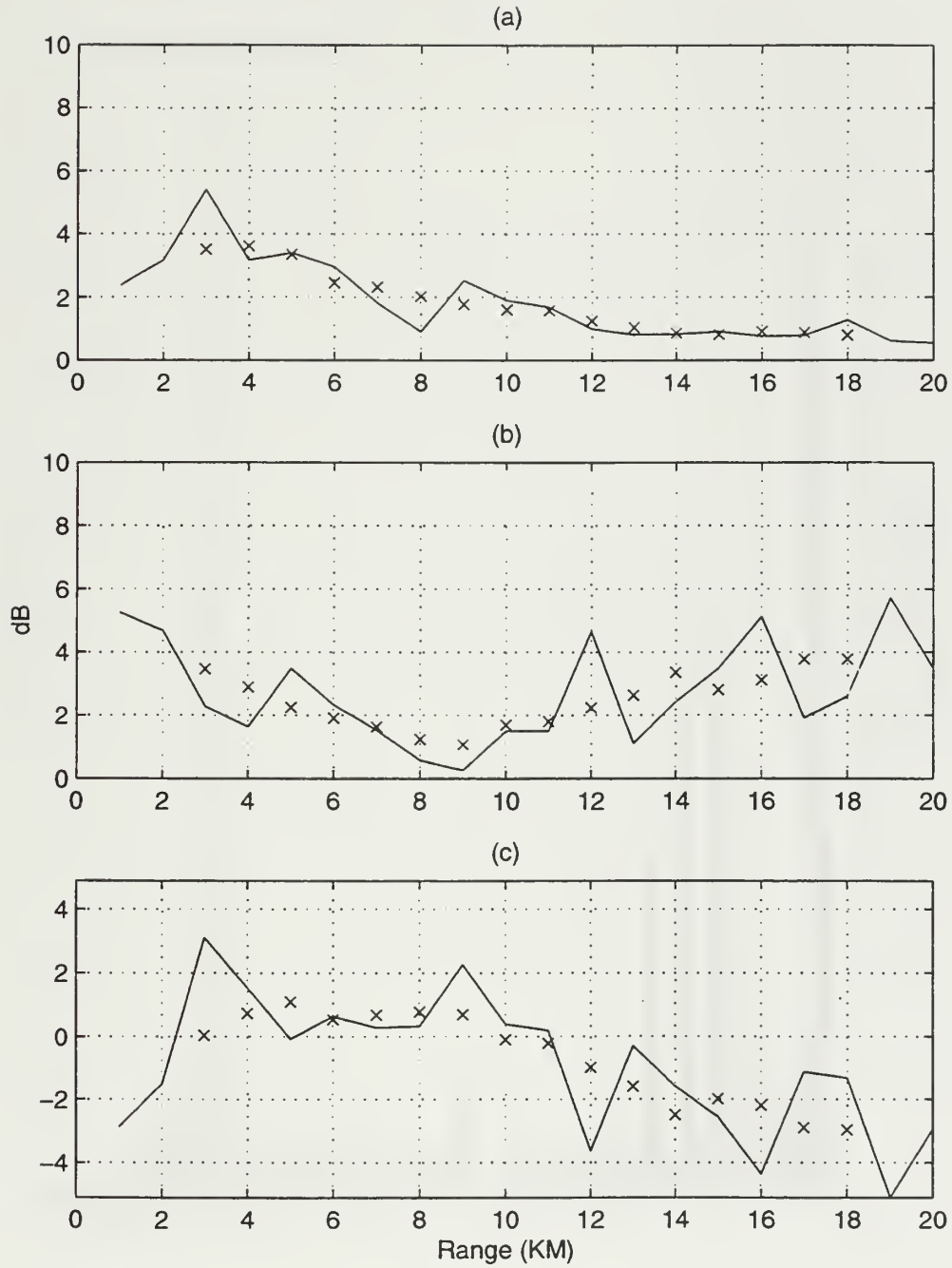

Figure 46. (a) ESL plotted for run 21 with a target depth at 11 m. (b) ESL plotted for run 21 with a target depth at 77 m. (c) The ESL difference between targets at 11 and 77 m. (Graph properties as before)

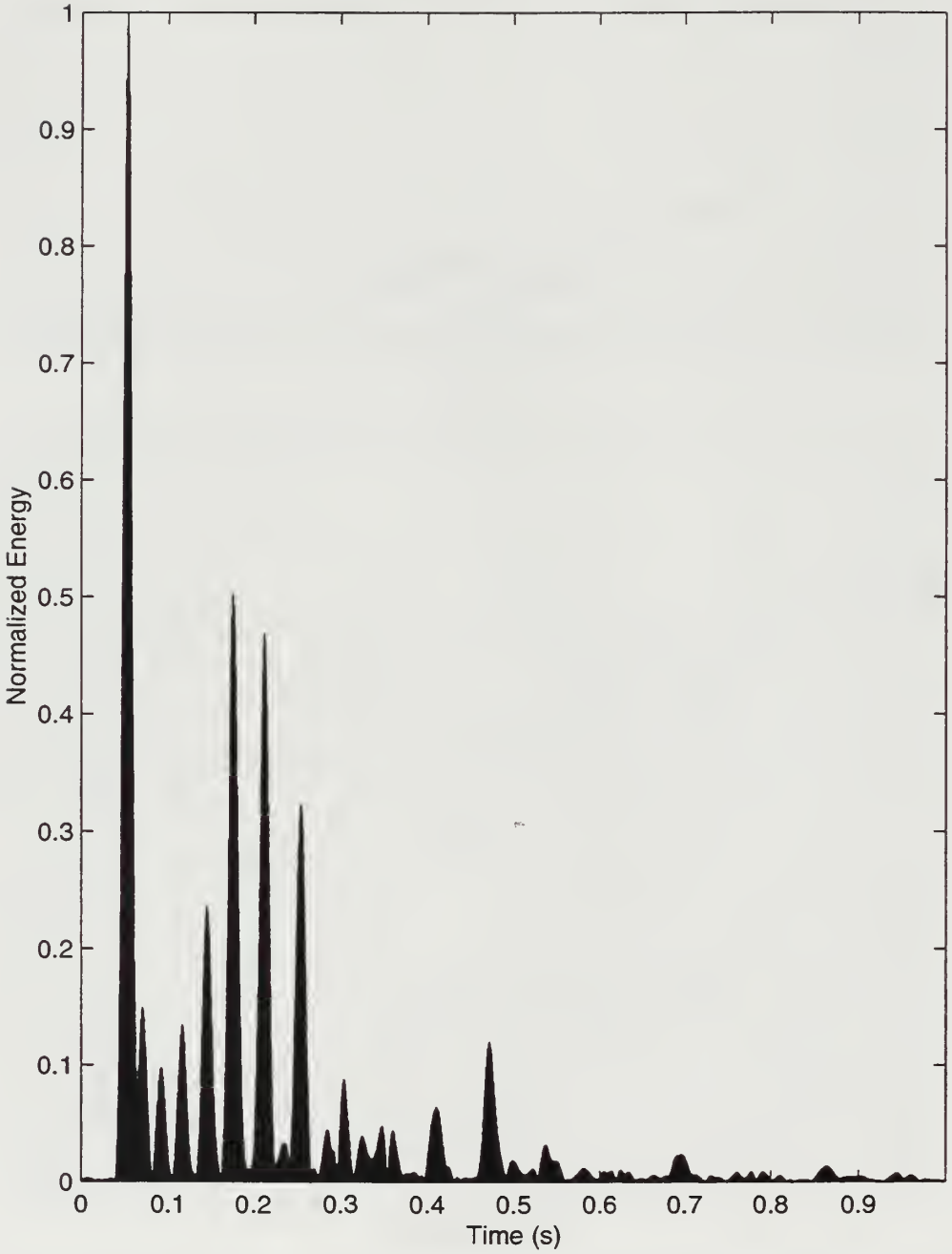

Figure 47. The graph shows the time domain information for run 21 at 9 km with a target depth of 11 m.

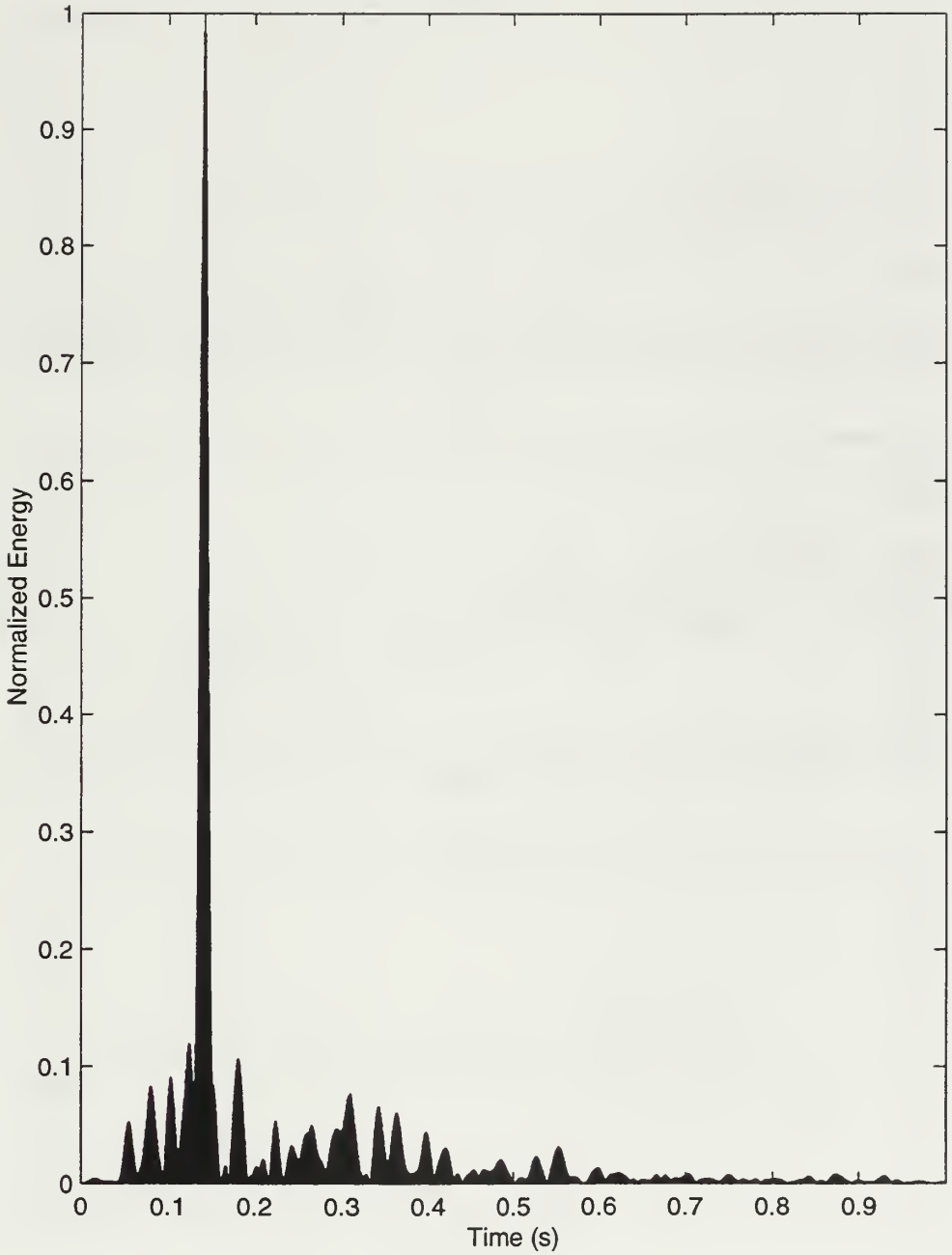

Figure 48. The graph shows the time domain information for run 21 at 9 km with a target depth of 77 m.

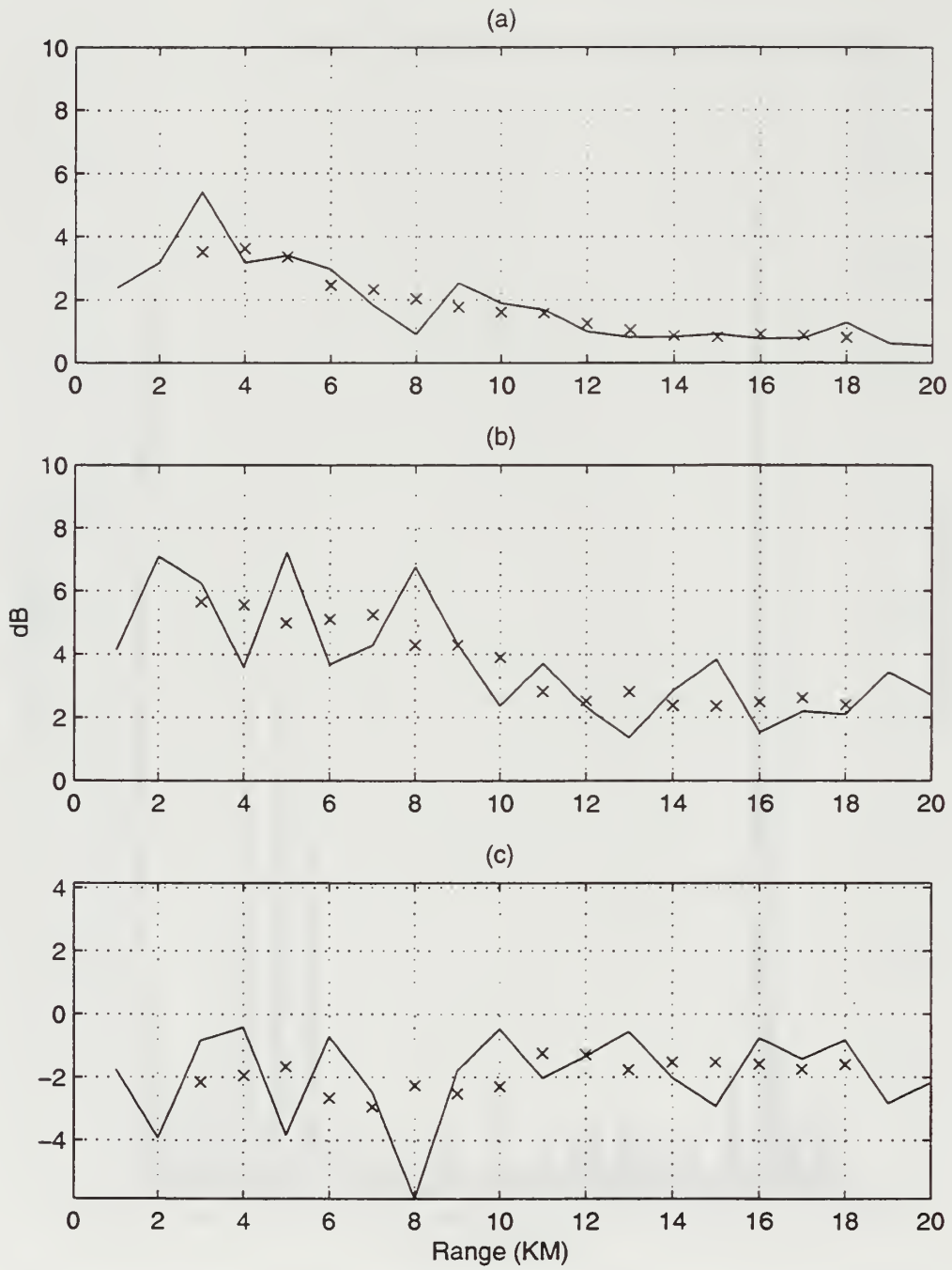

Figure 49. (a) ESL plotted for run 21 with a target depth at 11 m. (b) ESL plotted for run 21 with a target depth at 99 m. (c) The ESL difference between targets at 11 and 99 m. (Graph properties as before)

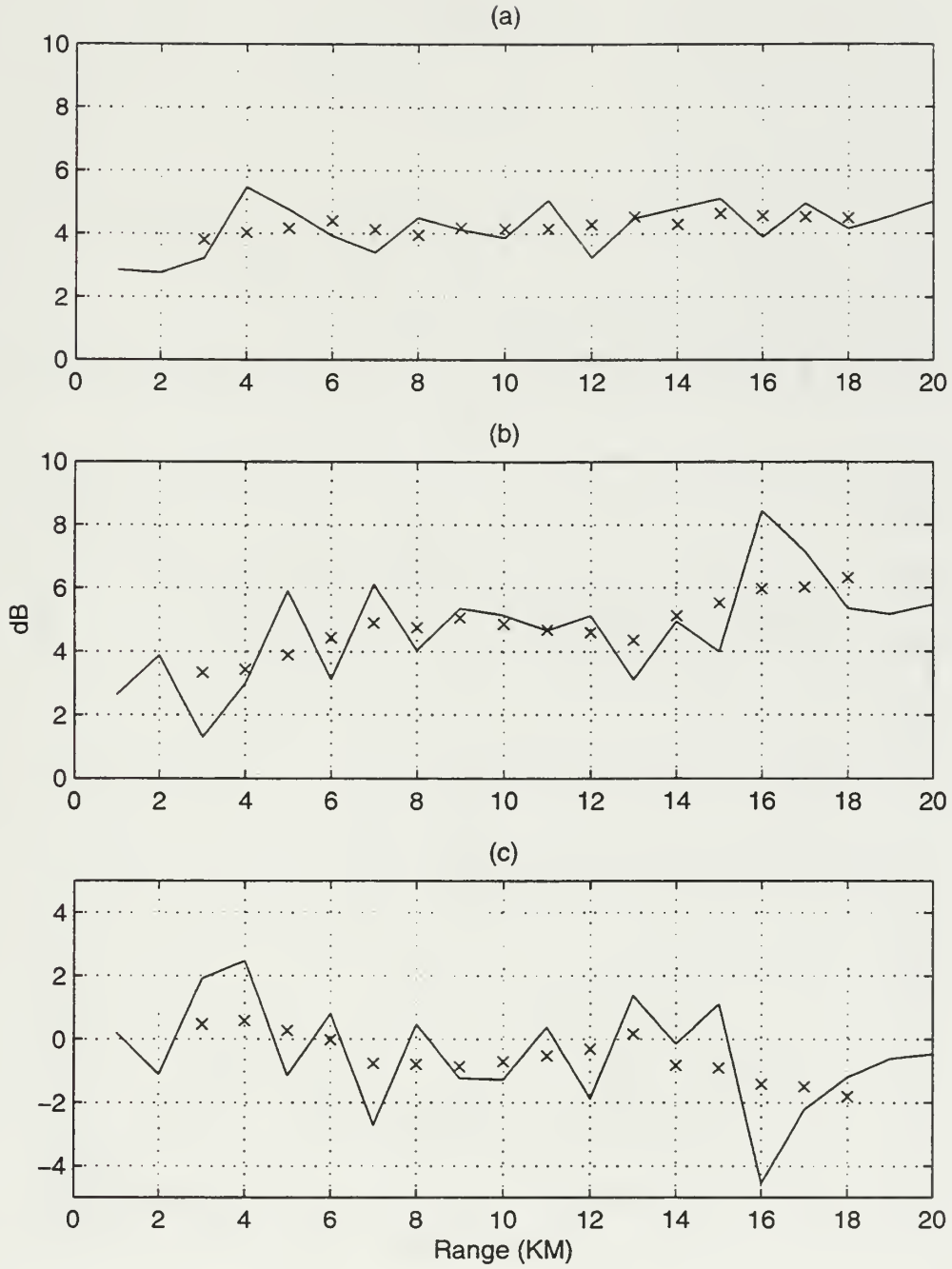

Figure 50. (a) ESL plotted for run 23 with a target depth at 11 m. (b) ESL plotted for run 23 with a target depth at 33 m. (c) The ESL difference between targets at 11 and 33 m. (Graph properties as before)

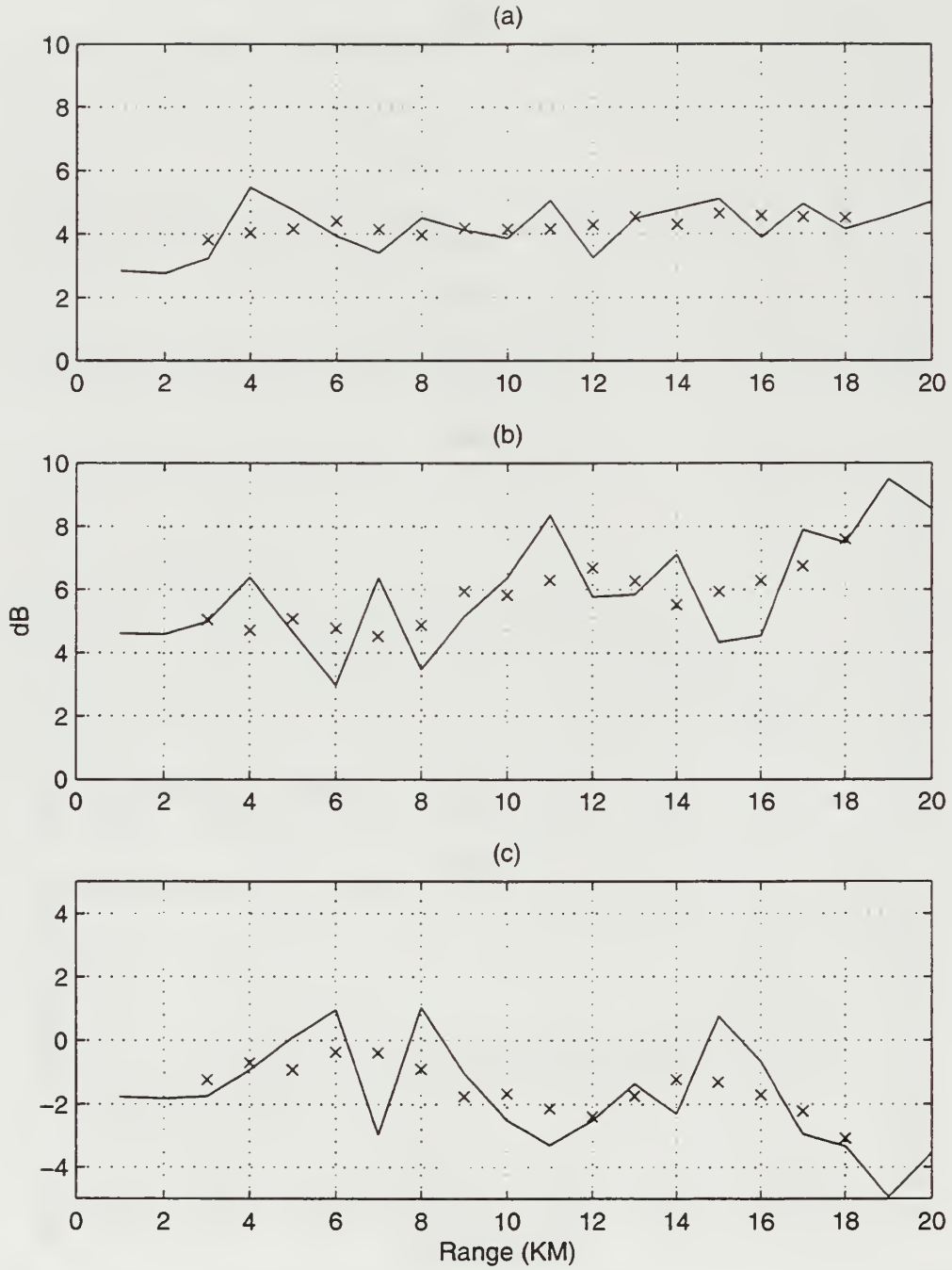

Figure 51. (a) ESL plotted for run 23 with a target depth at 11 m. (b) ESL plotted for run 23 with a target depth at 55 m. (c) The ESL difference between targets at 11 and 55 m. (Graph properties as before)

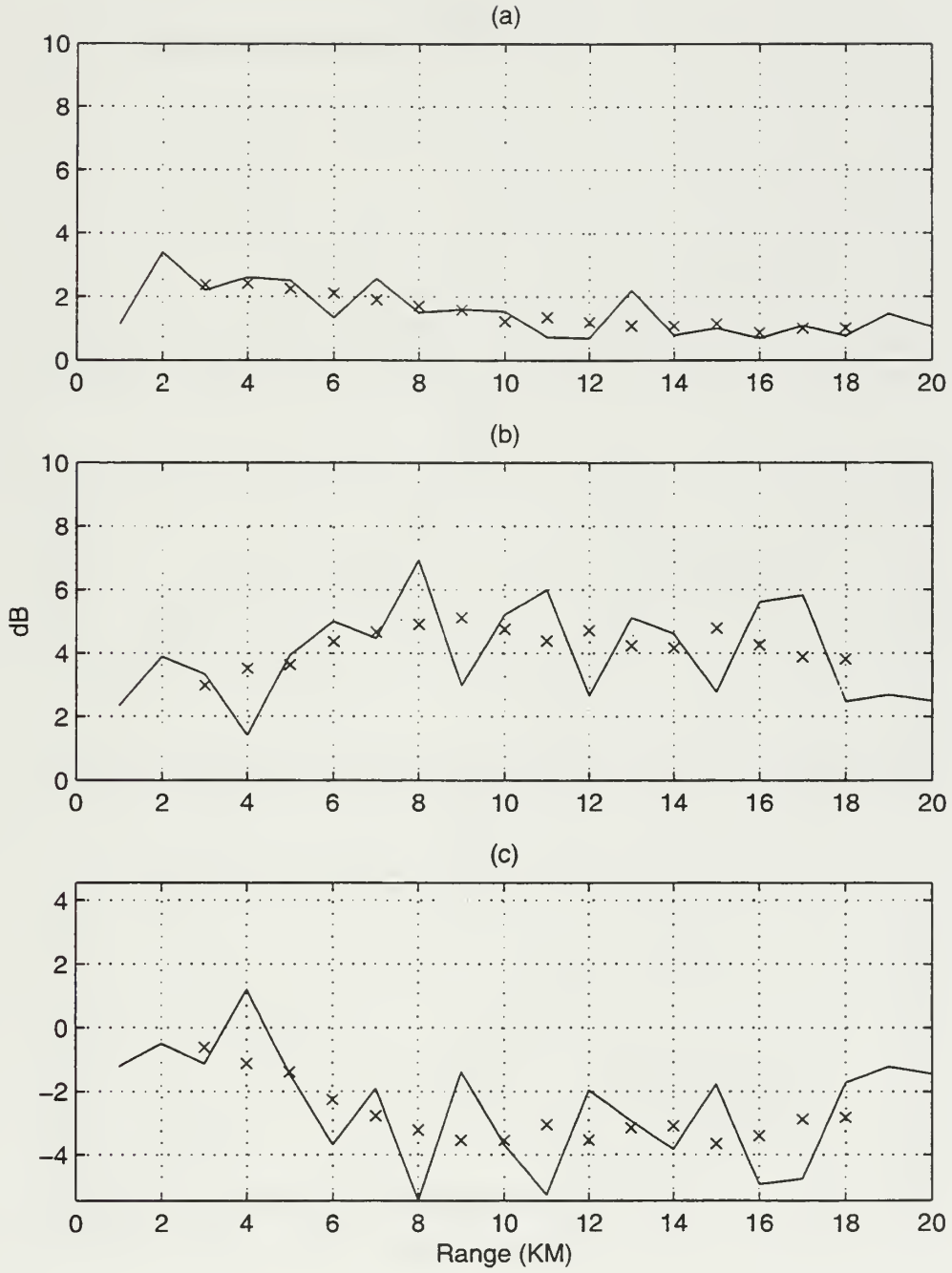

Figure 52. (a) ESL plotted for run 25 with a target depth at 11 m. (b) ESL plotted for run 25 with a target depth at 33 m. (c) The ESL difference between run targets at 11 and 33 m. (Graph properties as before)

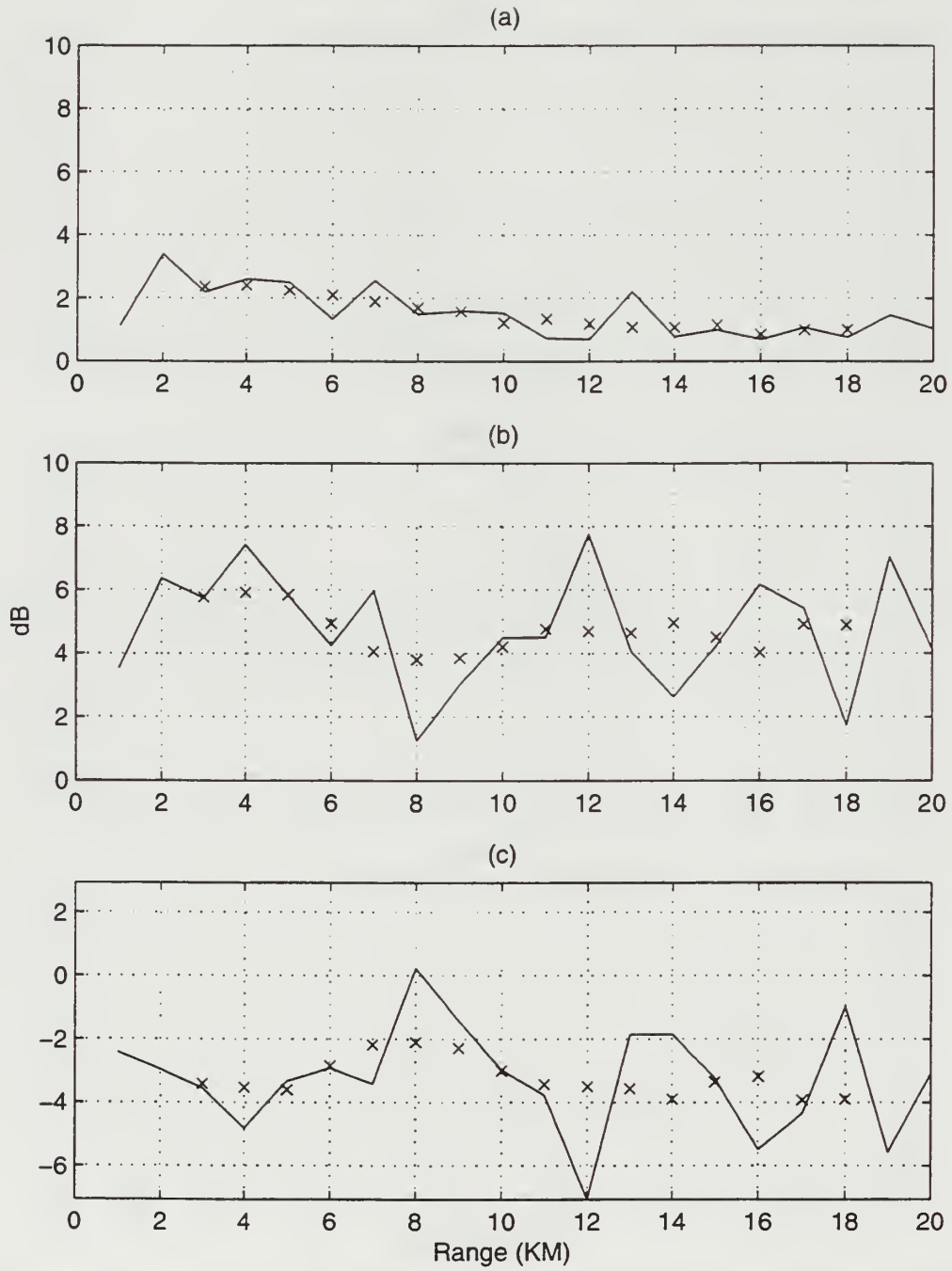

Figure 53. (a) ESL plotted for run 25 with a target depth at 11 m. (b) ESL plotted for run 25 with a target depth at 99 m. (c) The ESL difference between targets at 11 and 99 m. (Graph properties as before)

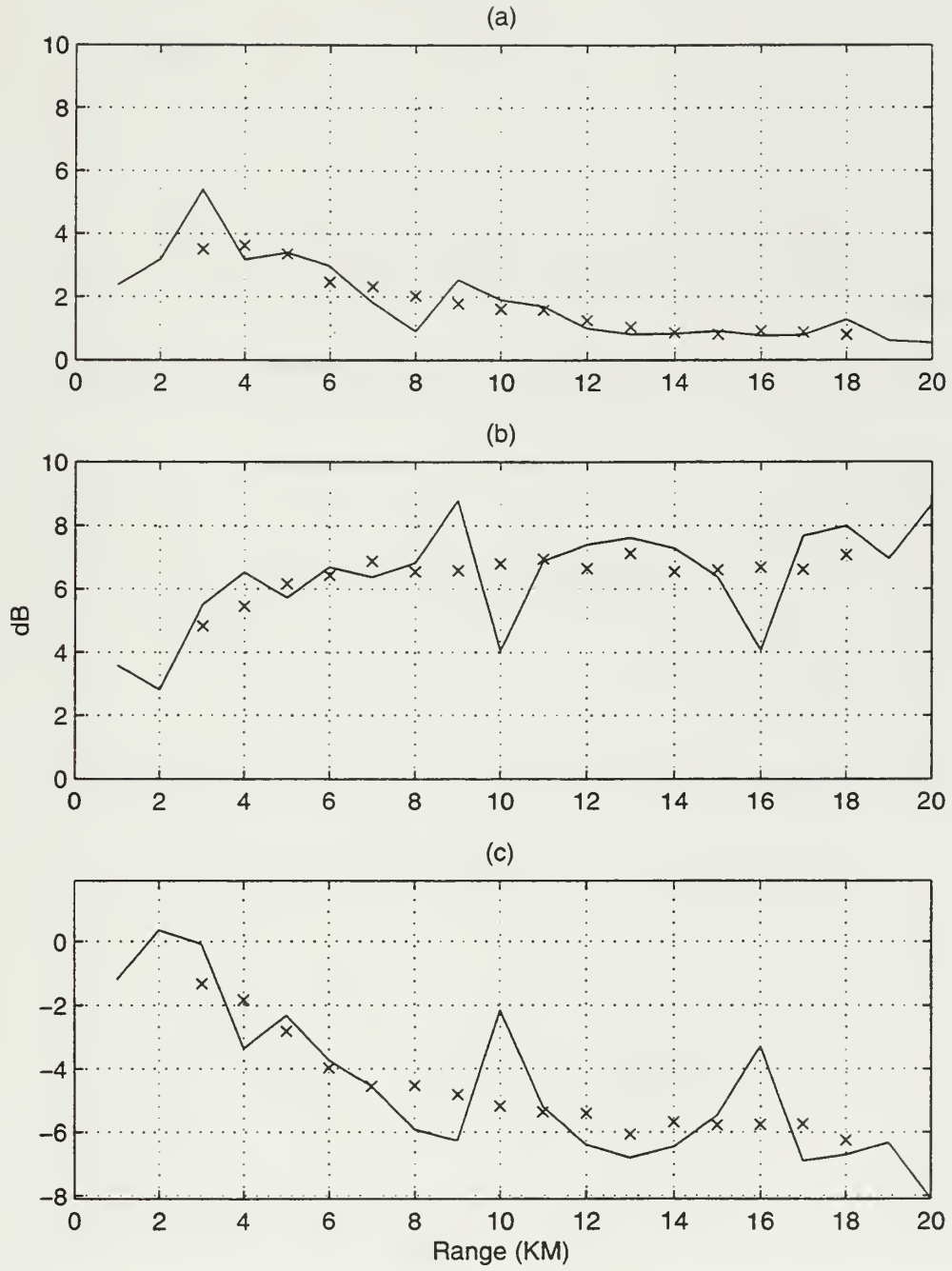

Figure 54. (a) ESL plotted for run 21 with a source and target depth of 11 m each. (b) ESL plotted for run 30 with a source and target depth of 55 and 11 m respectively. (c) The ESL difference between run 21 and run 30. (Graph properties as before)

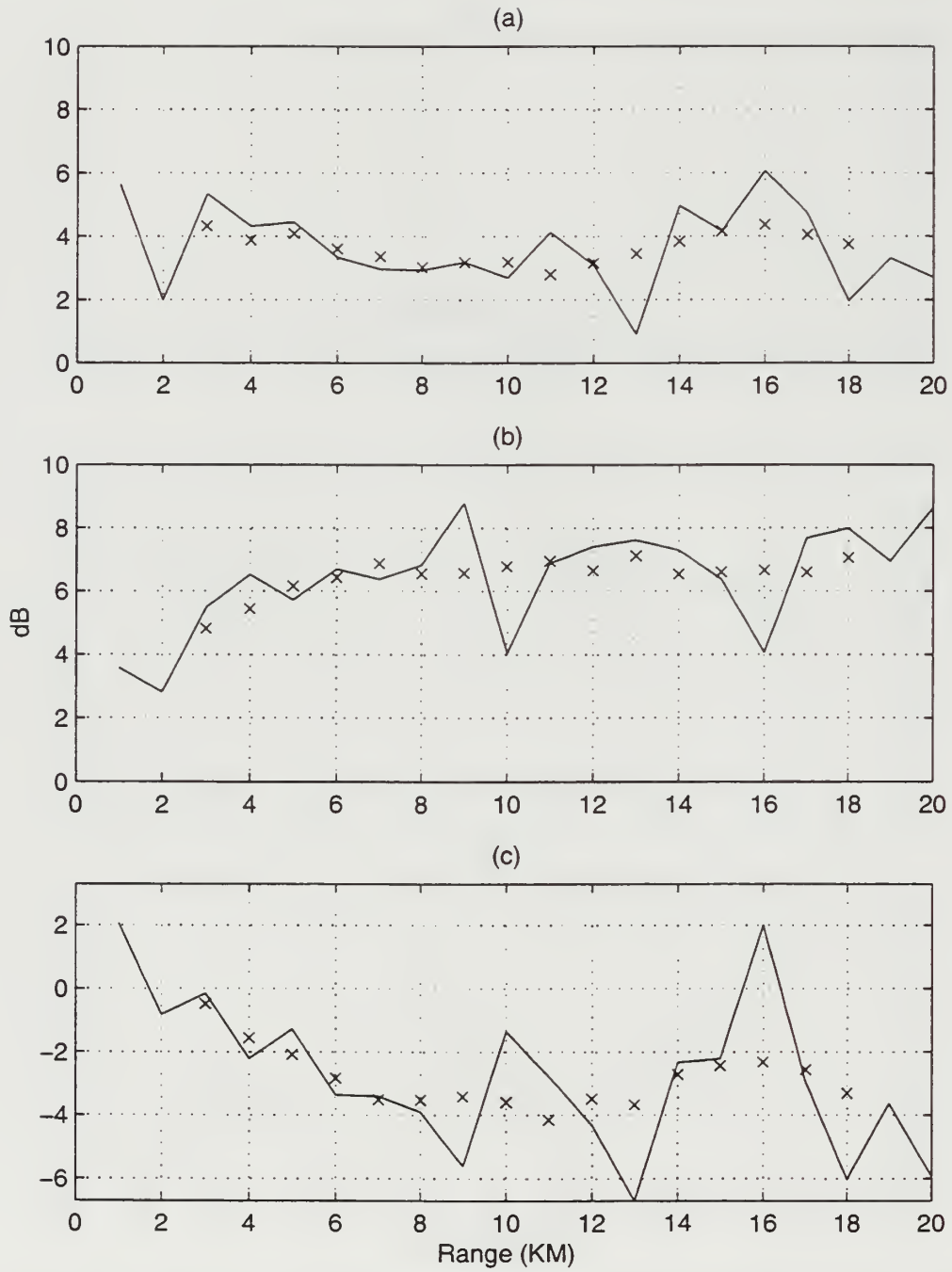

Figure 55. (a) ESL plotted for run 30 with a source and target depth of 55 m each. (b) ESL plotted for run 30 with a source and target depth of 55 and 11 m, respectively. (c) The ESL difference between target depths of 55 and 11 m. (Graph properties as before)

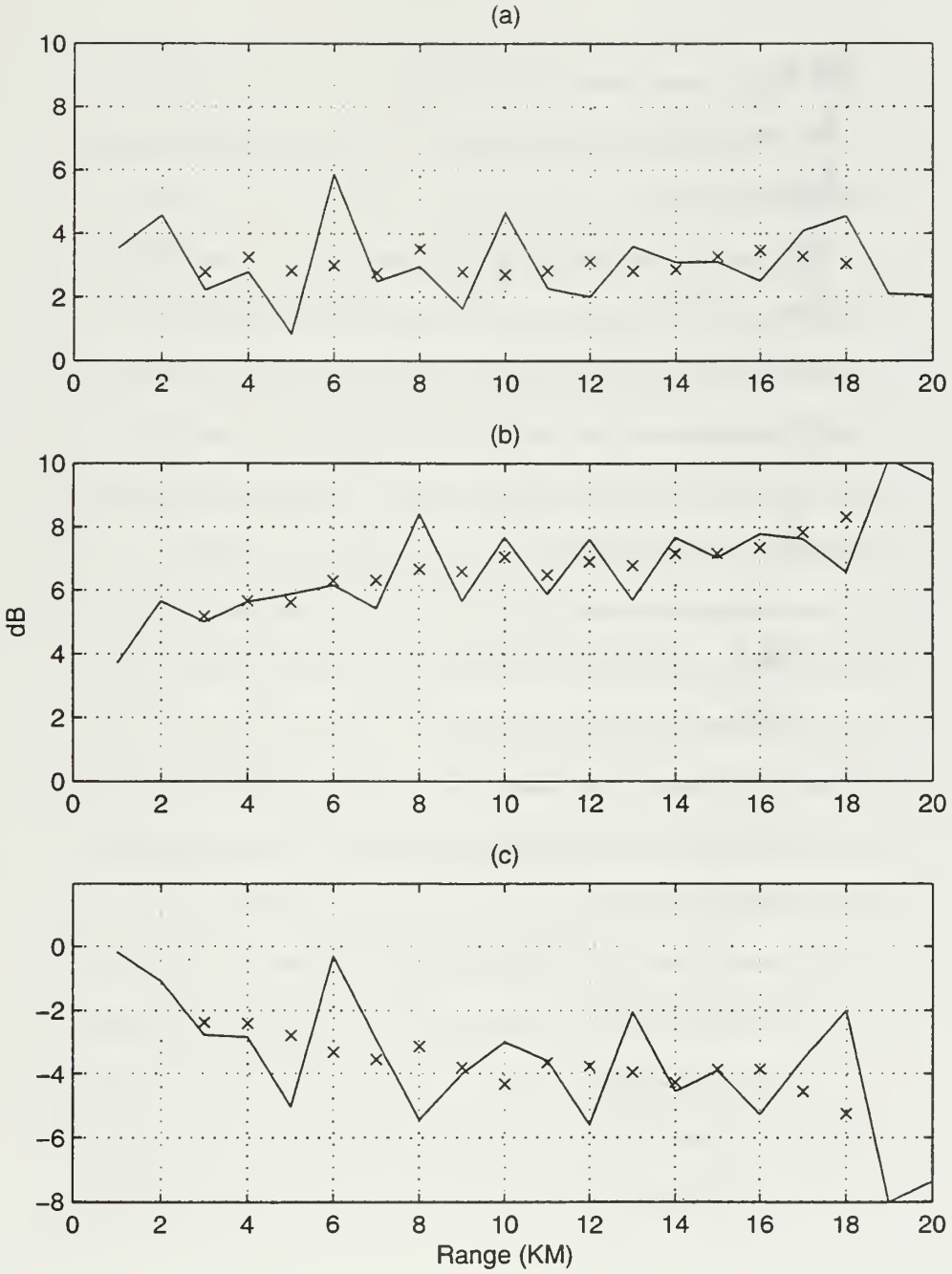

Figure 56. (a) ESL plotted for run 31 with a source and target depth of 99 m each. (b) ESL plotted for run 31 with a source and target depth of 99 and 11 m respectively. (c) The ESL difference between targets at 99 and 11 m. (Graph properties as before)

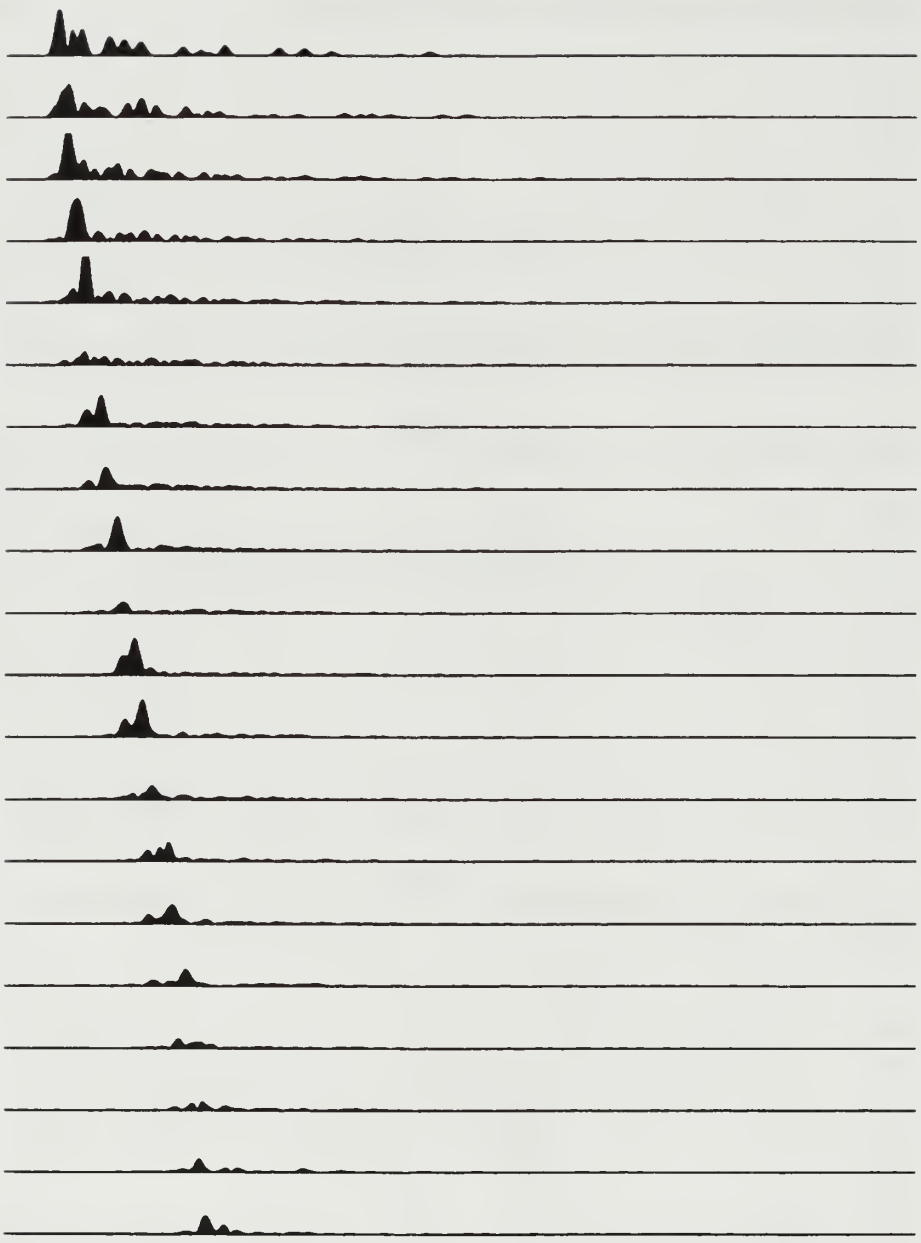

Figure 57. A plot of output pulses in the time domain. The graph starts with 1 km at the top to 20 km at the bottom in increments of 1 km per graph. The pulses are derived from OTF's from run 31 with both source and target at 99 m.

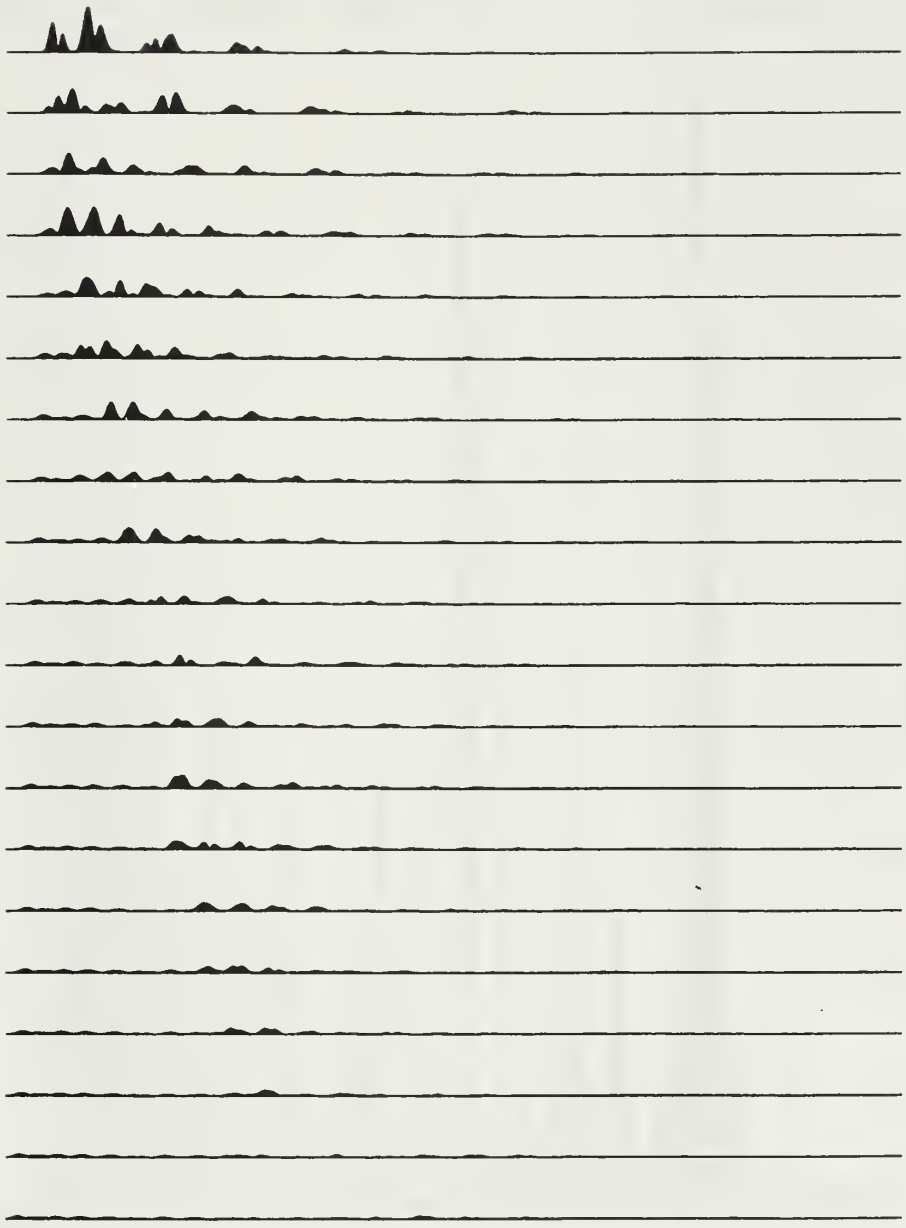

Figure 58. A plot of output pulses in the time domain. The graph starts with 1 km at the top to 20 km at the bottom in increments of 1 km per graph. The pulses are derived from OTF's from run 31 with a source at 99 m and target at 11 m.

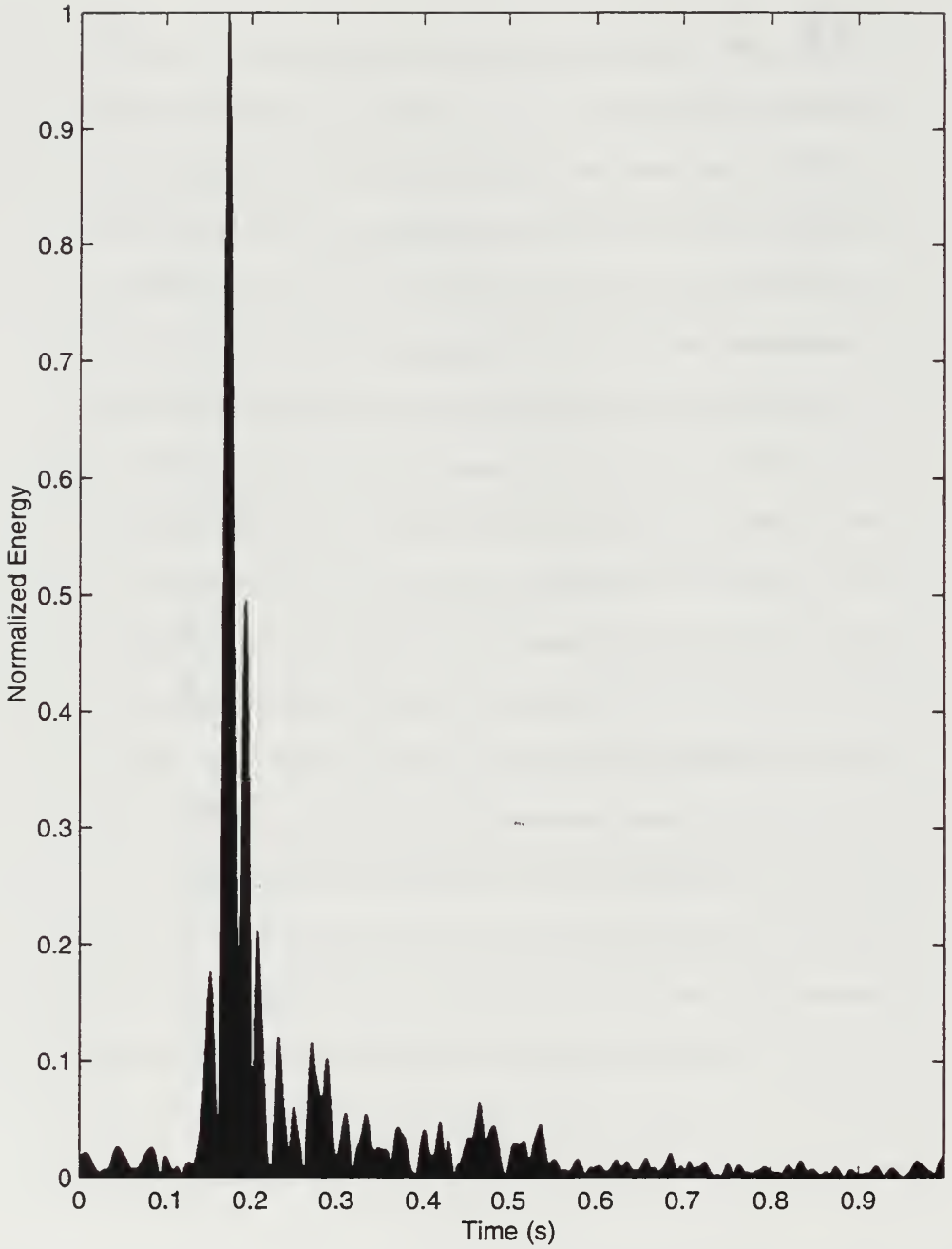

Figure 59. The graph shows the time domain information for run 31 at 20 km. The source and target are at 99 m depth.

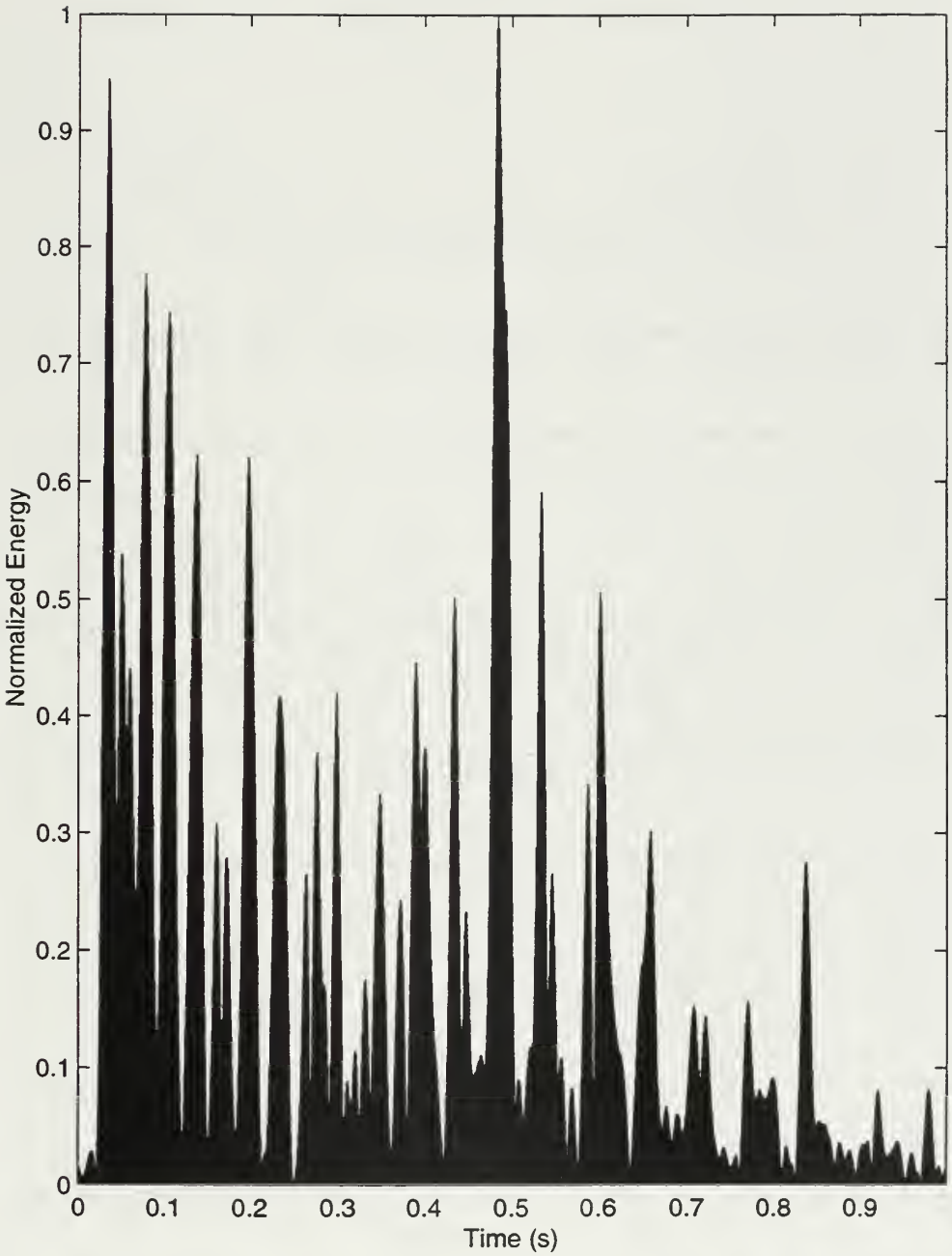

Figure 60. The graph shows the time domain information for run 31 at 20 km. The source is at 99 m, and the target is at 11 m depth.

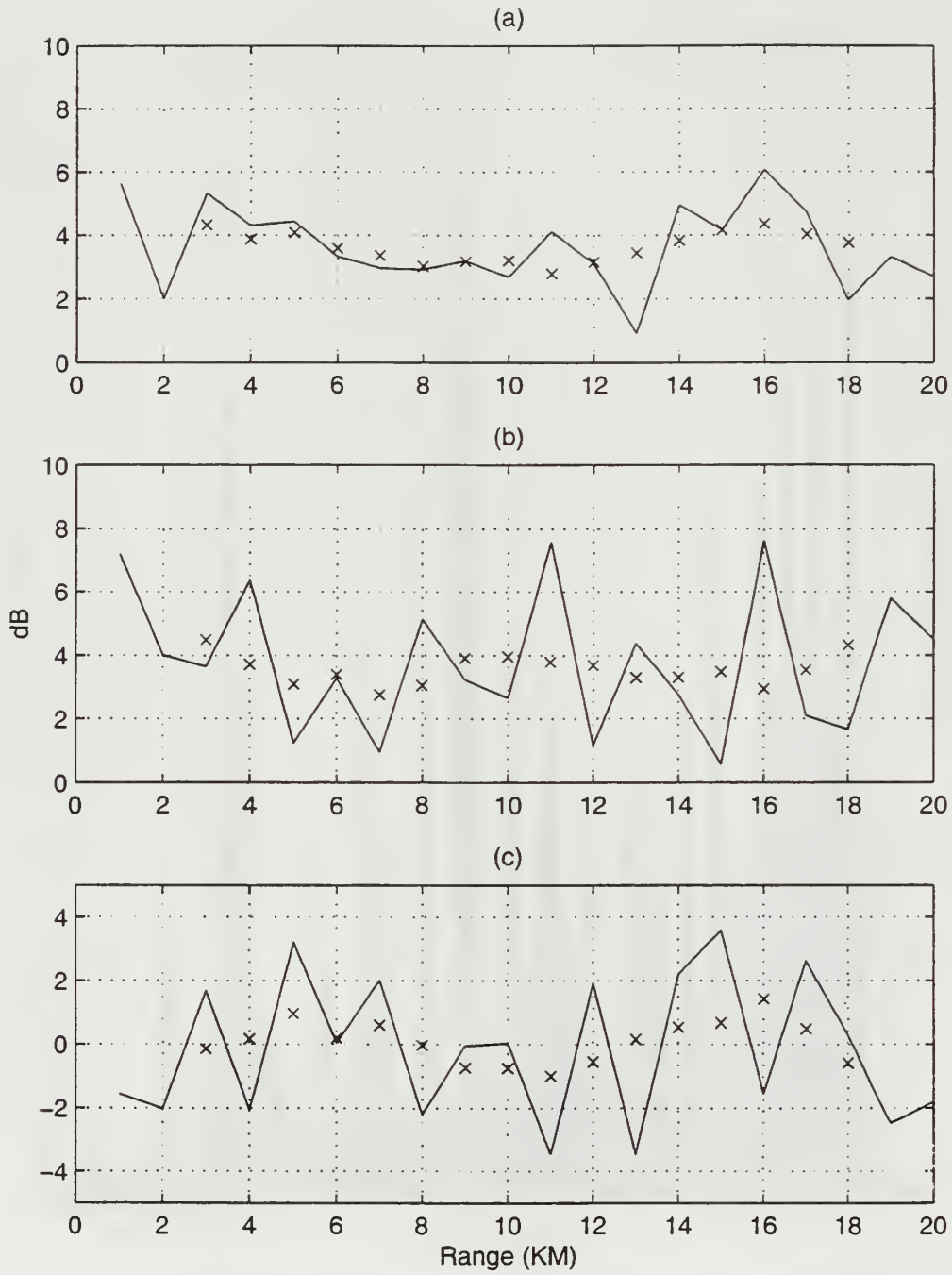

Figure 61. (a) ESL plotted for run 30 with a source and target depth of 55 m each. (b) ESL plotted for run 21 with a source and target depth of 11 and 55 m, respectively. (c) The ESL difference between run 30 and run 21. (Graph properties as before)

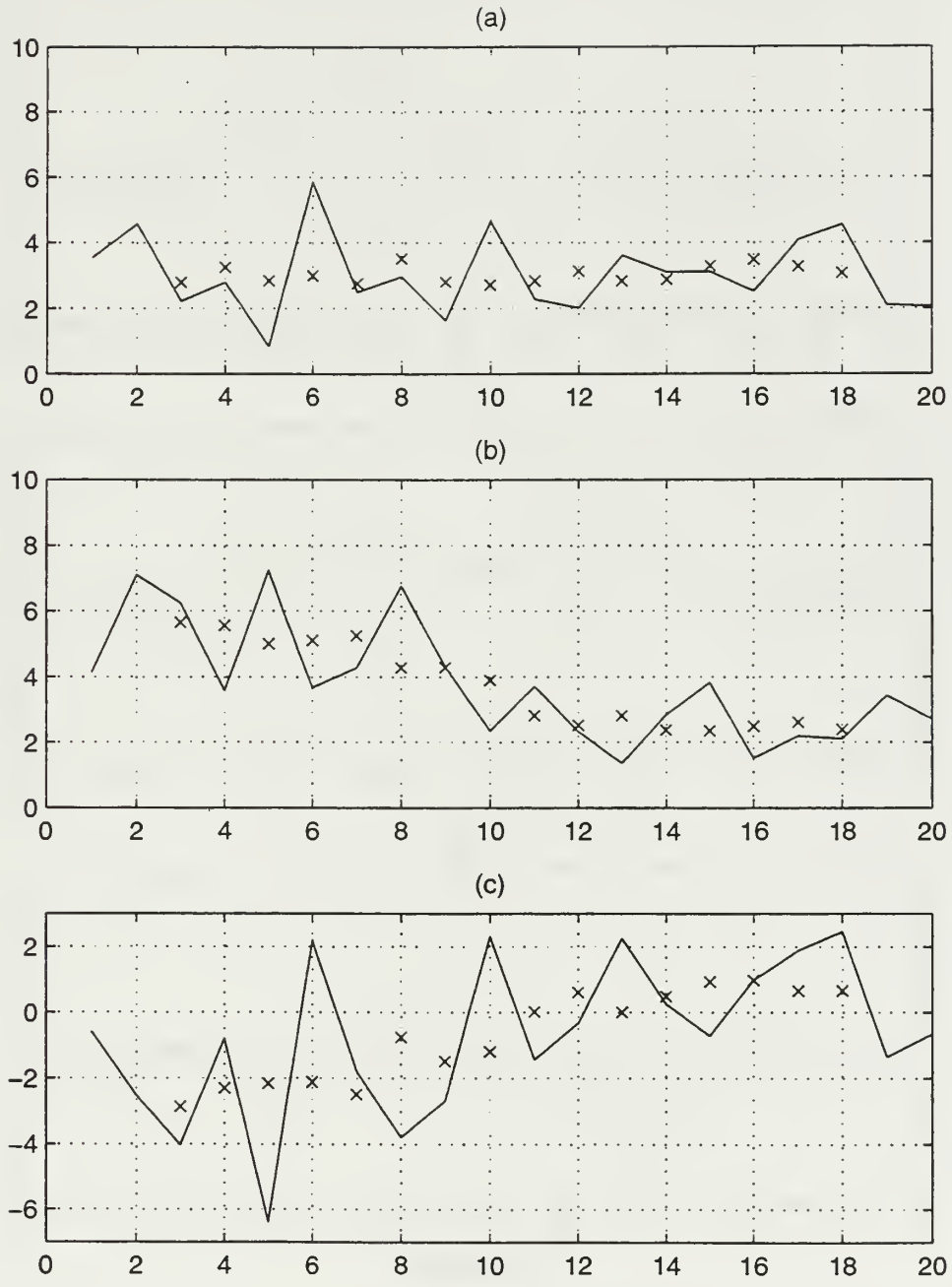

Figure 62. (a) ESL plotted for run 31 with a source and target depth of 99 m each. (b) ESL plotted for run 21 with a source and target depth of 11 and 99 m respectively. (c) The ESL difference between run 31 and run 21. (Graph properties as before)

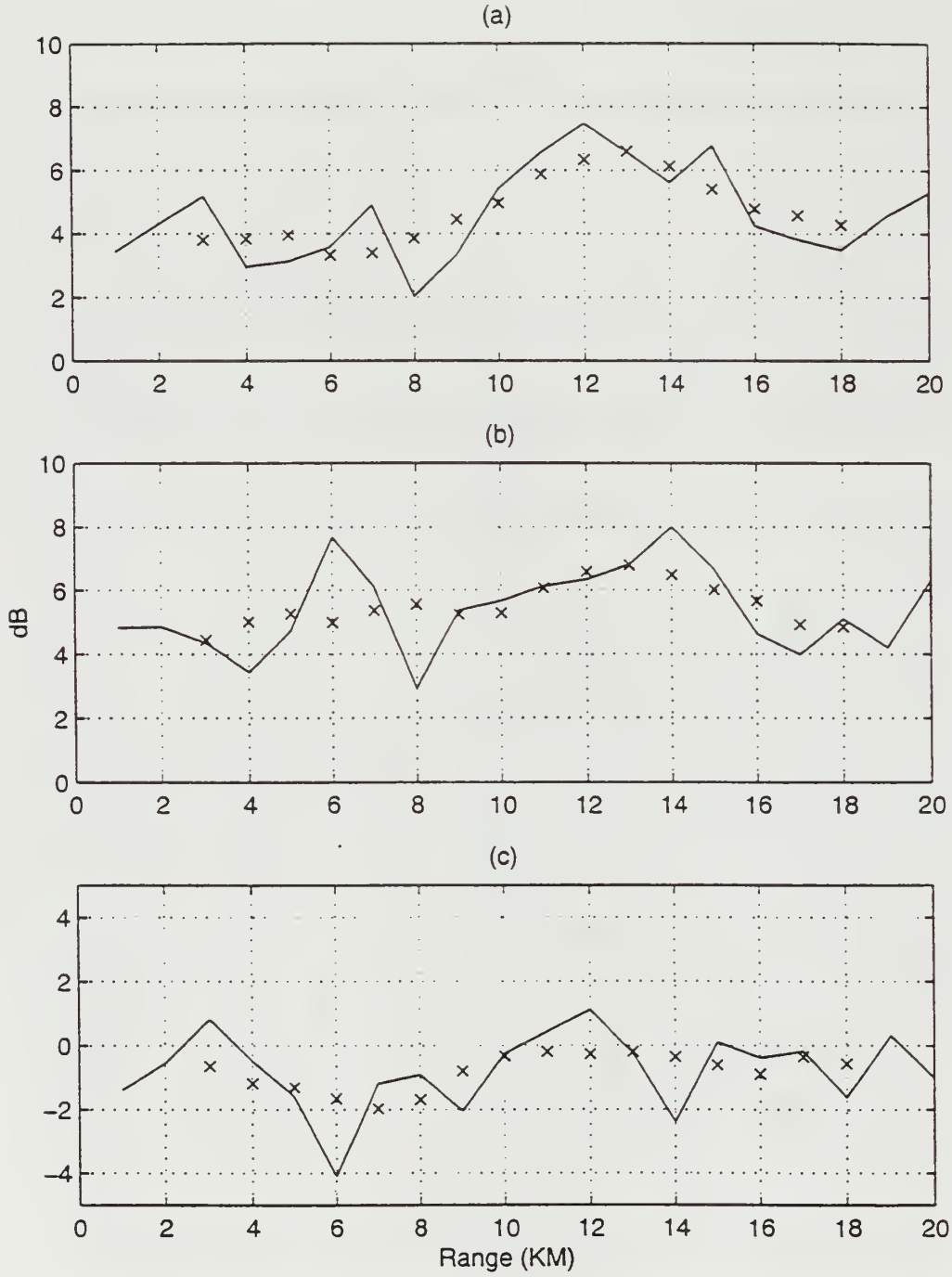

Figure 63. (a) ESL plotted for run 1 with a Blackman pulse. (b) ESL plotted for run 1 with a boxcar pulse. (c) The ESL difference between a Blackman pulse and a boxcar pulse. (Graph properties as before)

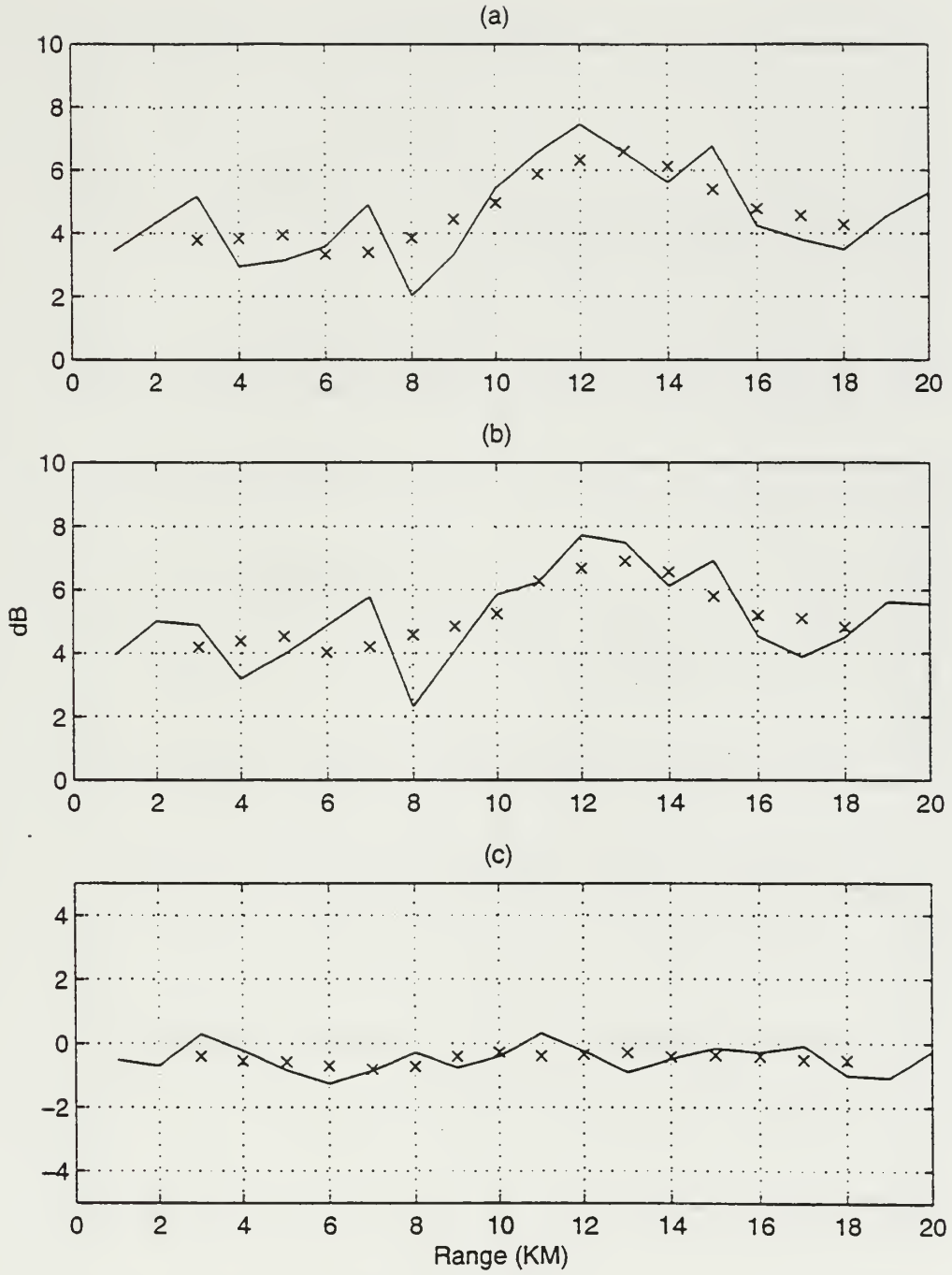

Figure 64. (a) ESL plotted for run 1 with a Blackman pulse. (b) ESL plotted for run 1 with a Hamming pulse. (c) The ESL difference between a Blackman pulse and a Hamming pulse. (Graph properties as before)

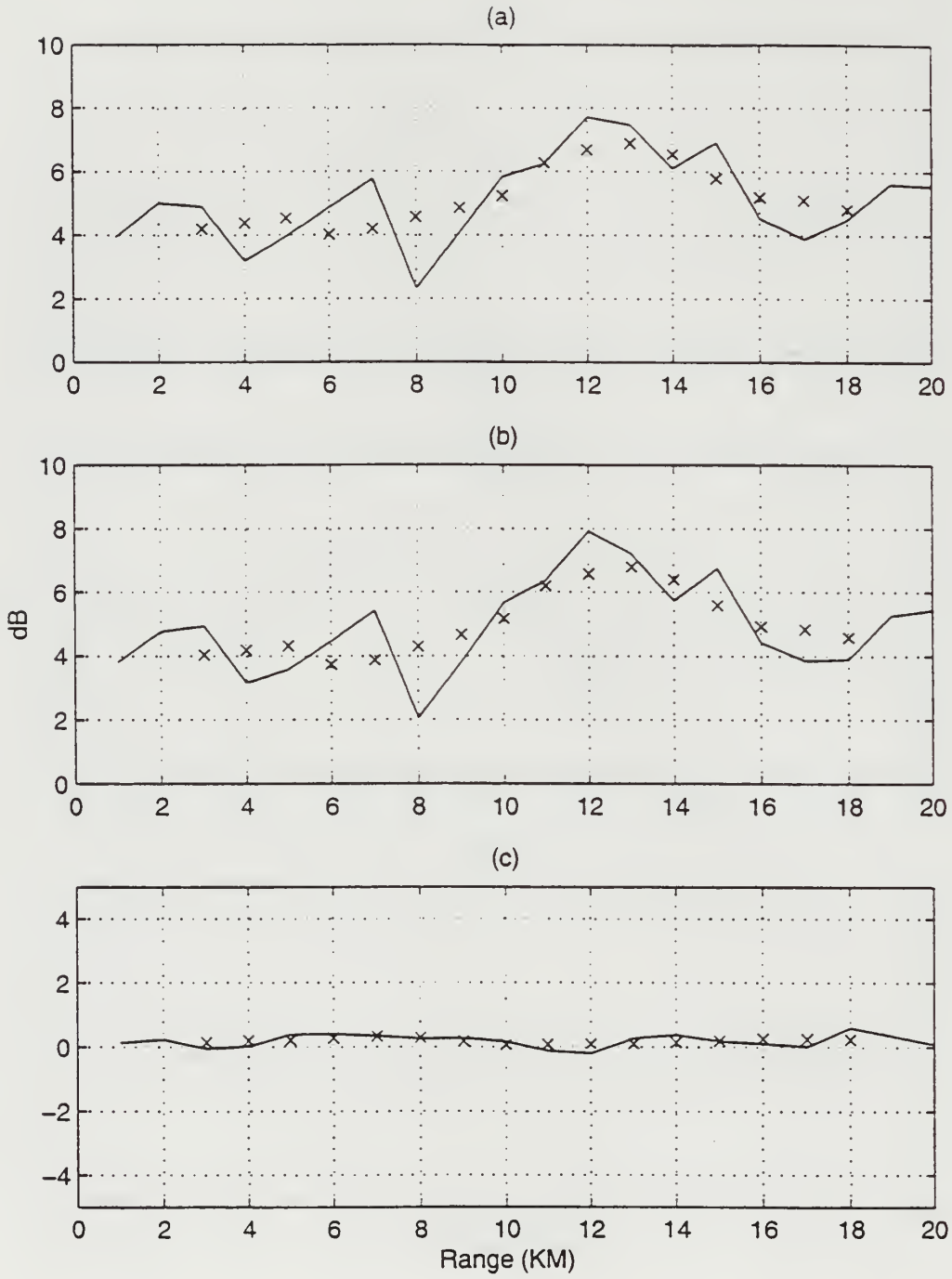

Figure 65. (a) ESL plotted for run 1 with a Hamming pulse. (b) ESL plotted for run 1 with a Hanning pulse. (c) The ESL difference between a Hamming pulse and a Hanning pulse. (Graph properties as before)

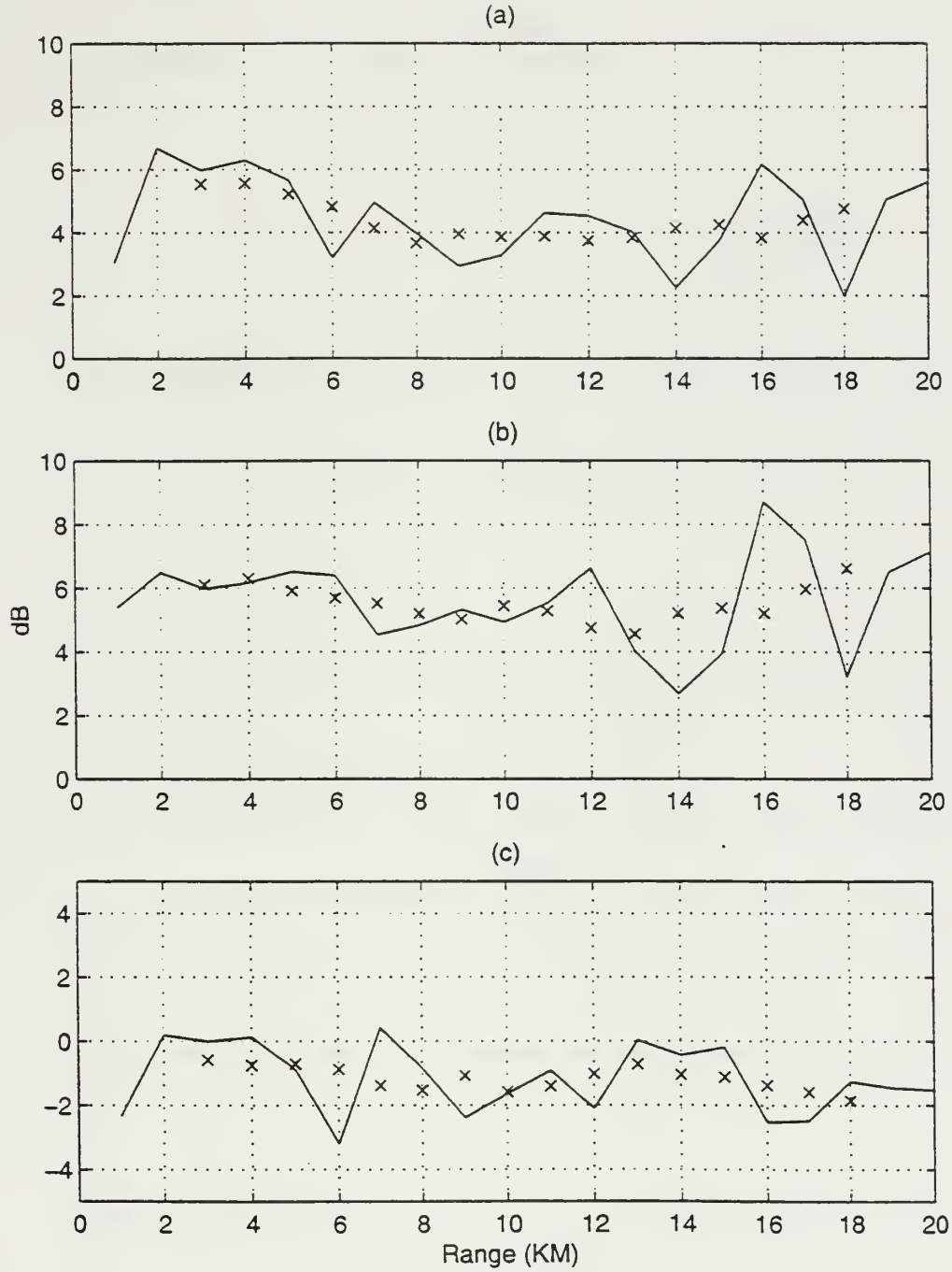

Figure 66. (a) ESL plotted for run 8 with a Blackman pulse. (b) ESL plotted for run 8 with a boxcar pulse. (c) The ESL difference between a Blackman pulse and a boxcar pulse. (Graph properties as before)

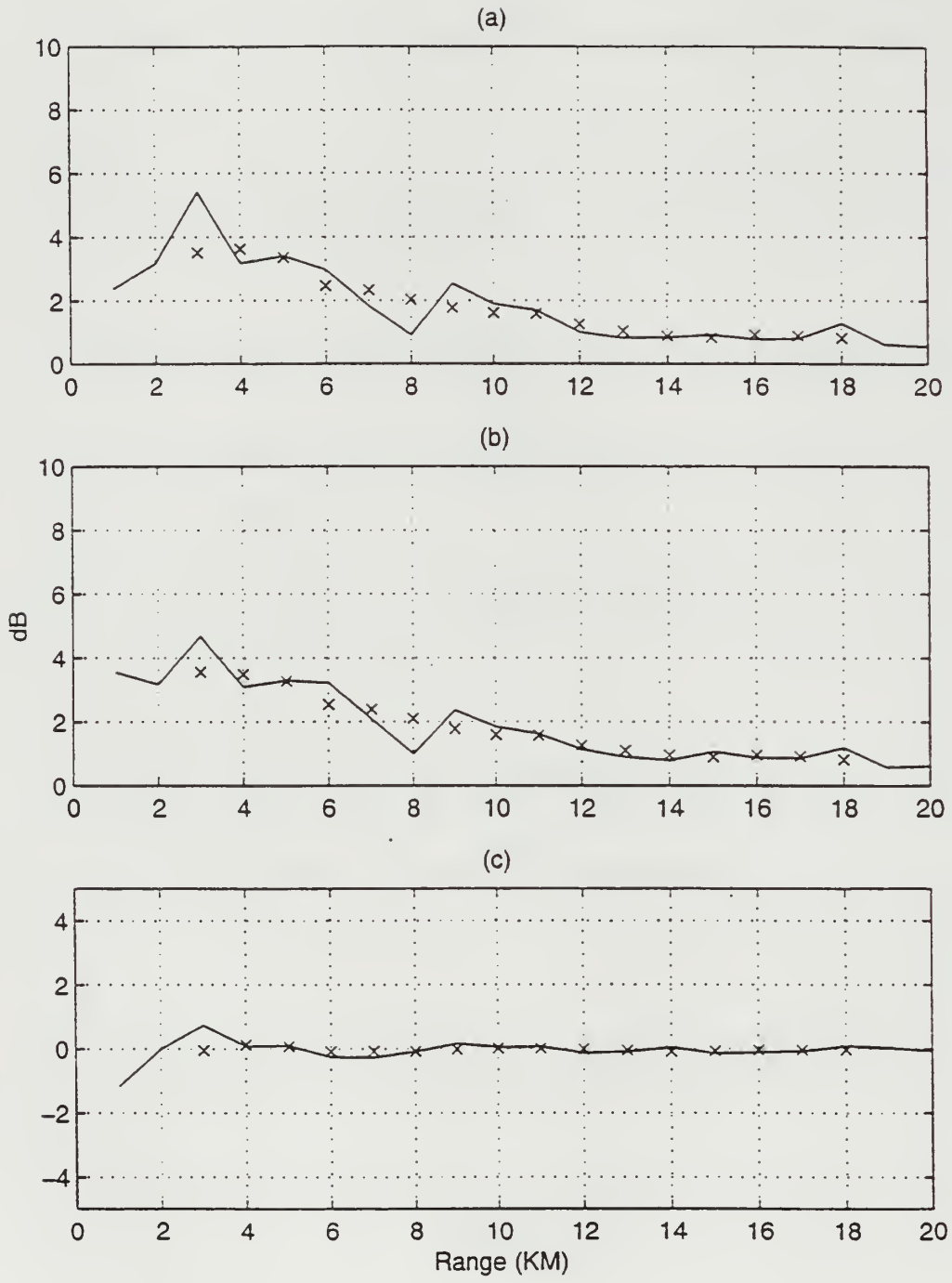

Figure 67. (a) ESL plotted for run 21 with a Blackman pulse. (b) ESL plotted for run 21 with a boxcar pulse. (c) The ESL difference between a Blackman pulse and a boxcar pulse. (Graph properties as before)

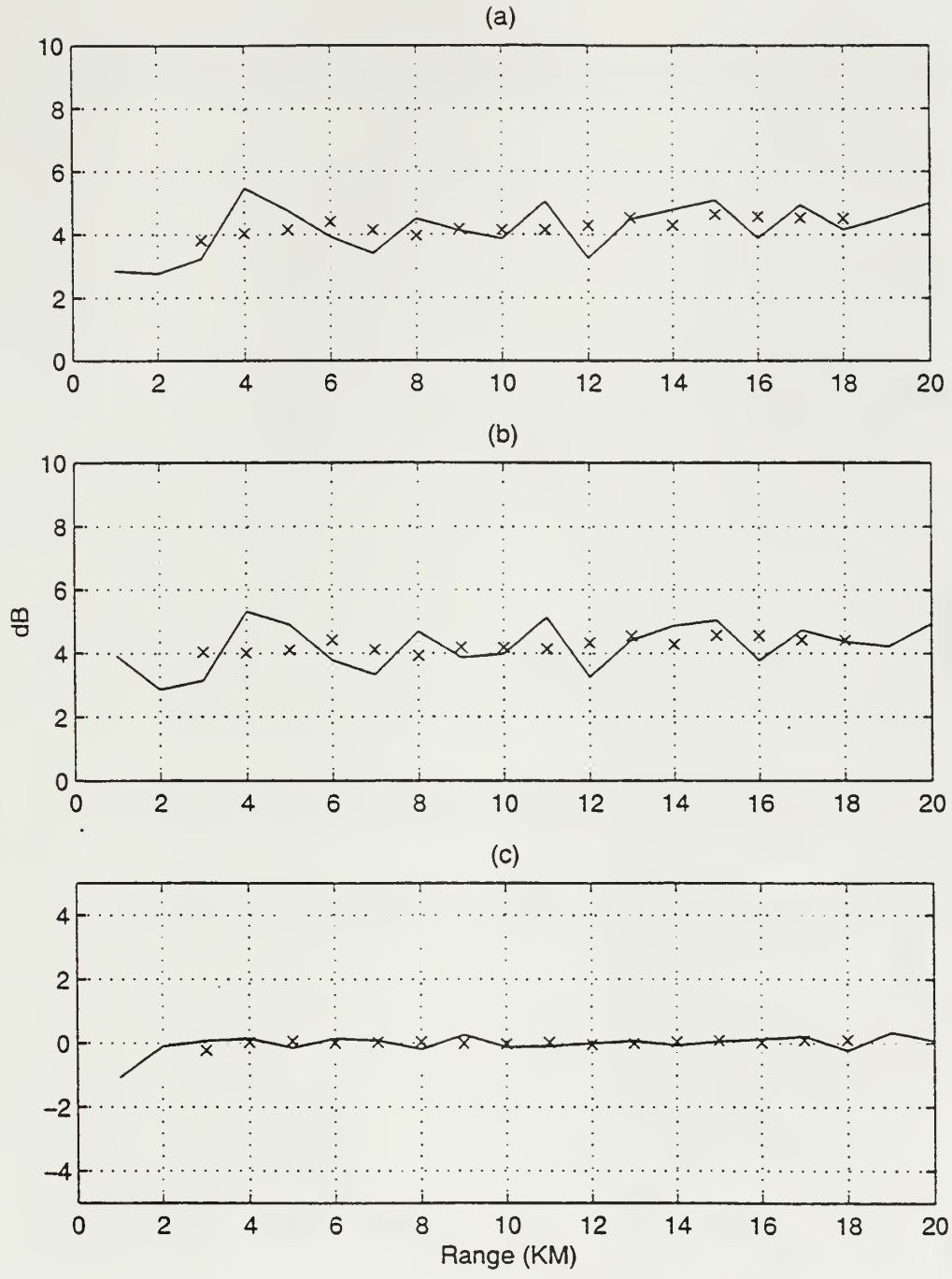

Figure 68. (a) ESL plotted for run 23 with a Blackman pulse. (b) ESL plotted for run 23 with a boxcar pulse. (c) The ESL difference between a Blackman pulse and a boxcar pulse. (Graph properties as before)

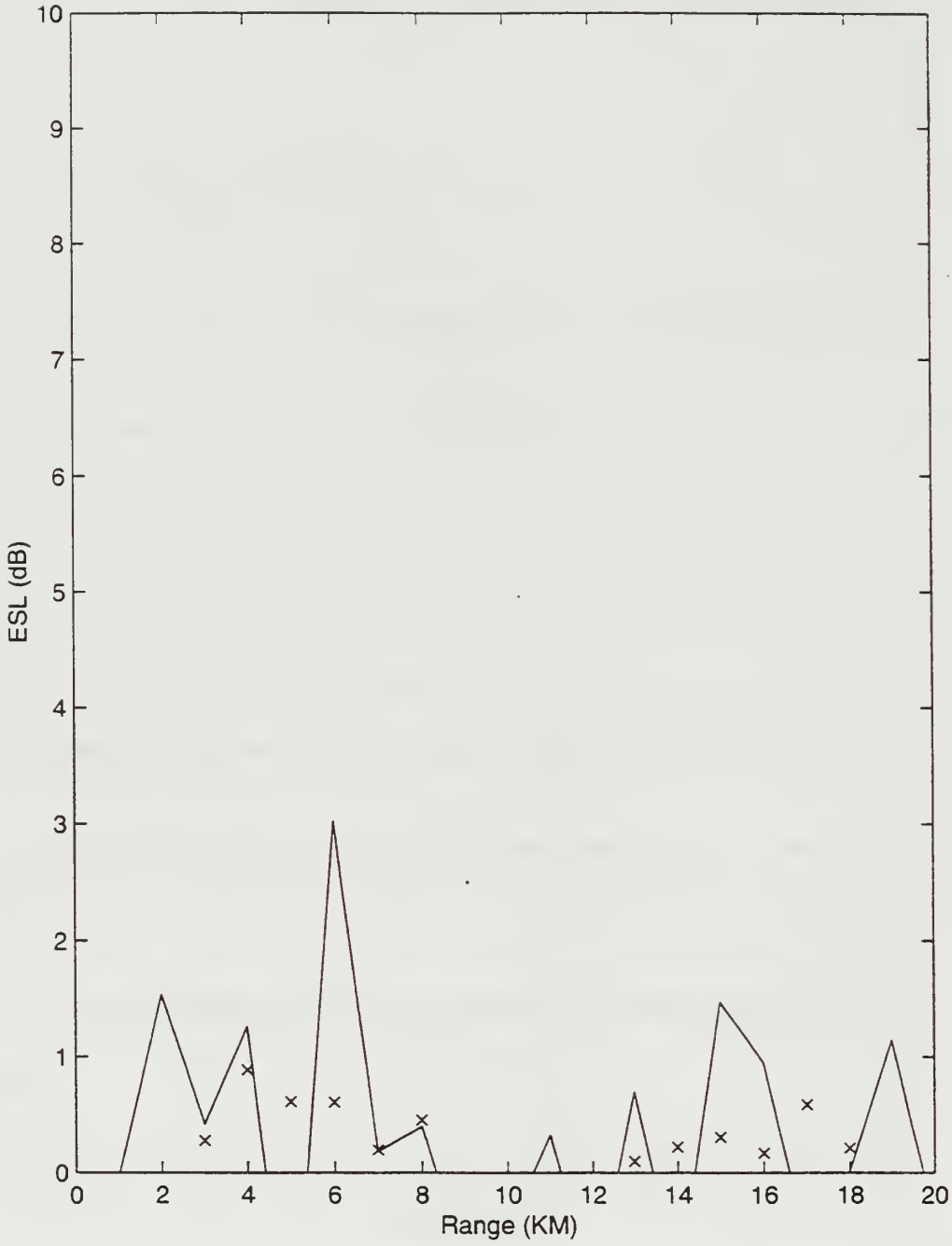

Figure 69. ESL plotted for run 1 with a 20 millisecond LFM pulse.

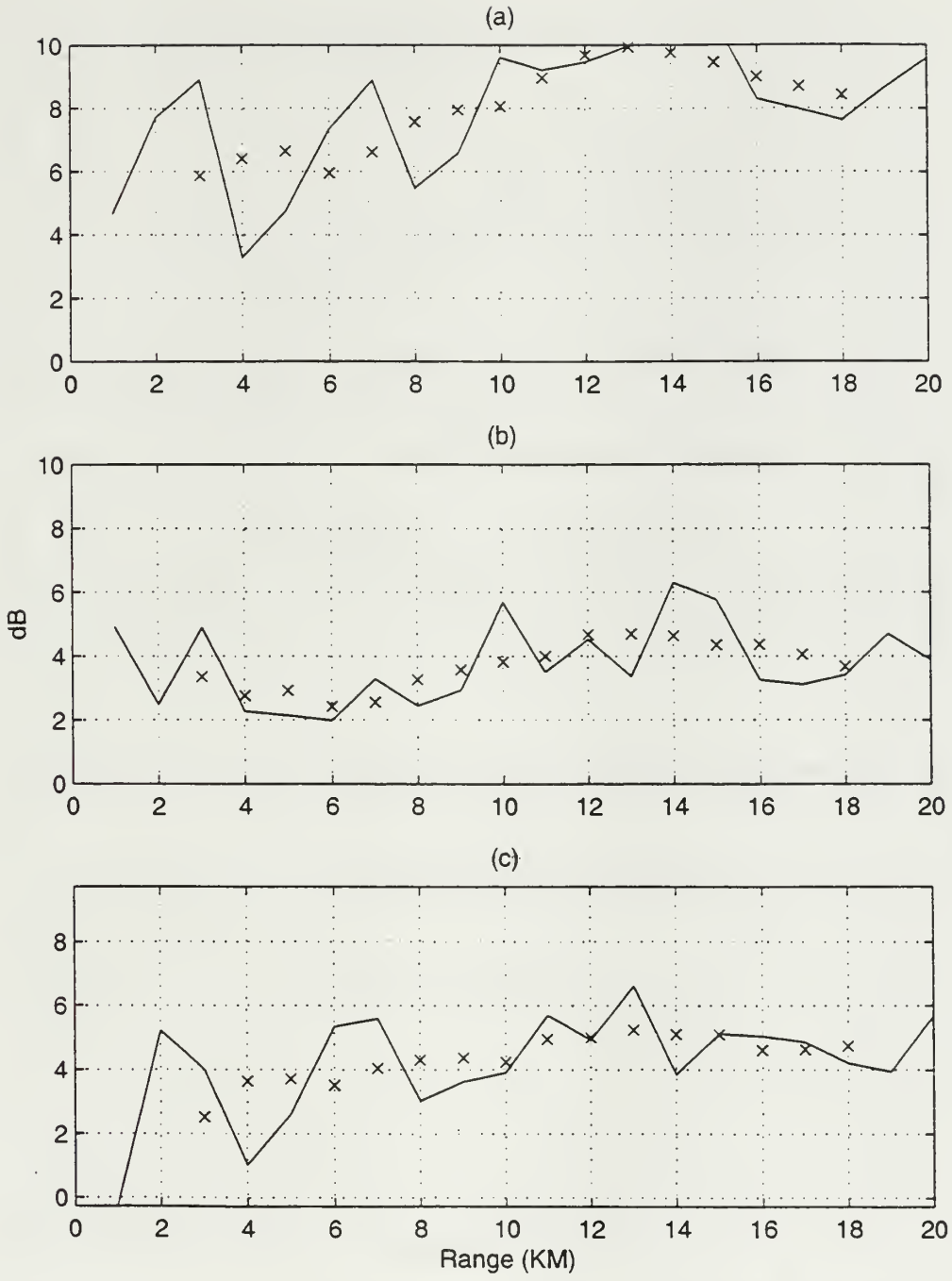

Figure 70. (a) MML plotted for run 1 with a Blackman pulse. (b) MML plotted for run 1 with a 20 millisecond CW pulse. (c) The MML difference between a Blackman pulse and a CW pulse. (Graph properties as before)

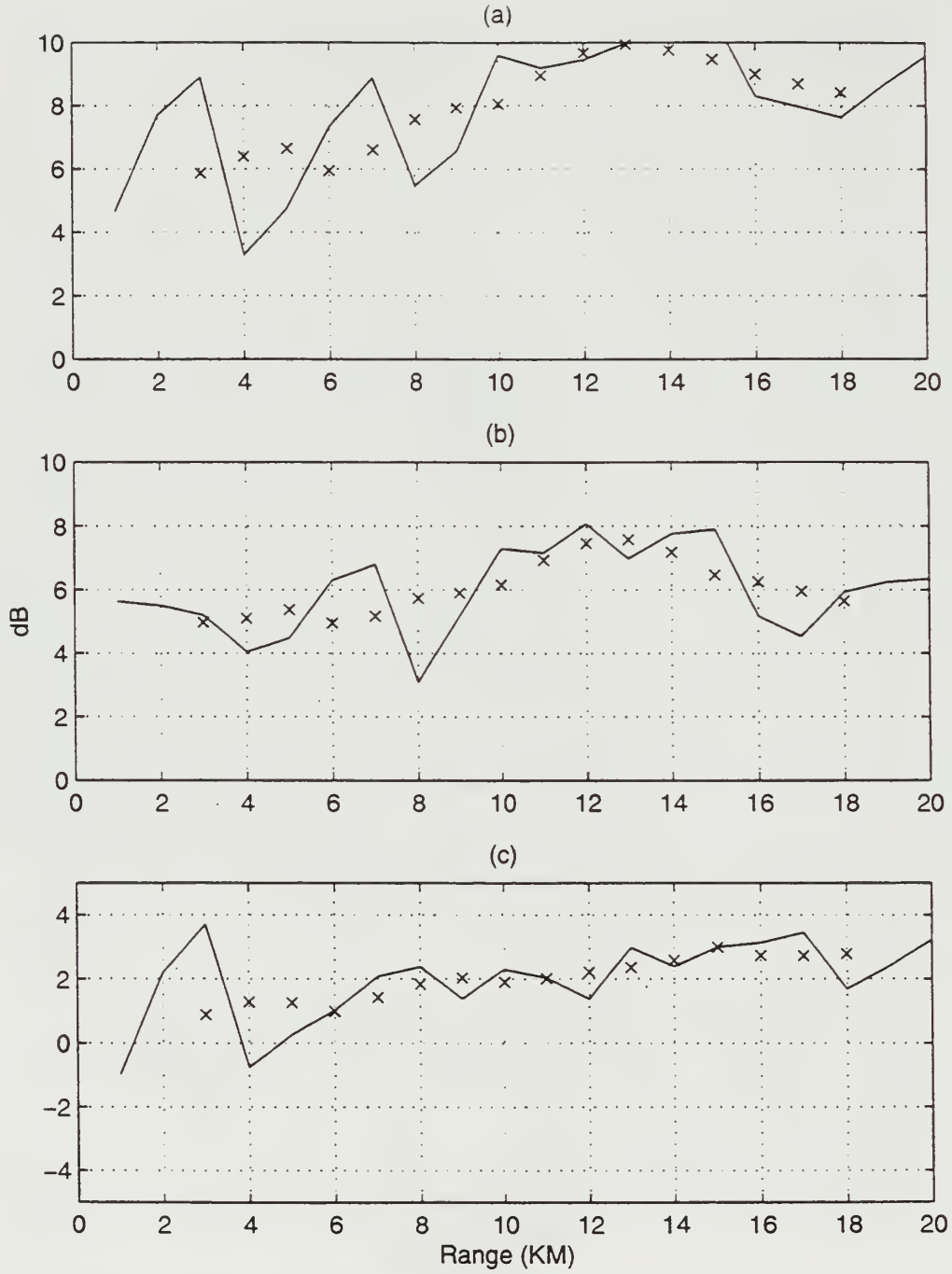

Figure 71. (a) MML plotted for run 1 with a Blackman pulse. (b) MML plotted for run 1 with a 20 millisecond LFM pulse. (c) The MML difference between a Blackman pulse and a LFM pulse. (Graph properties as before)

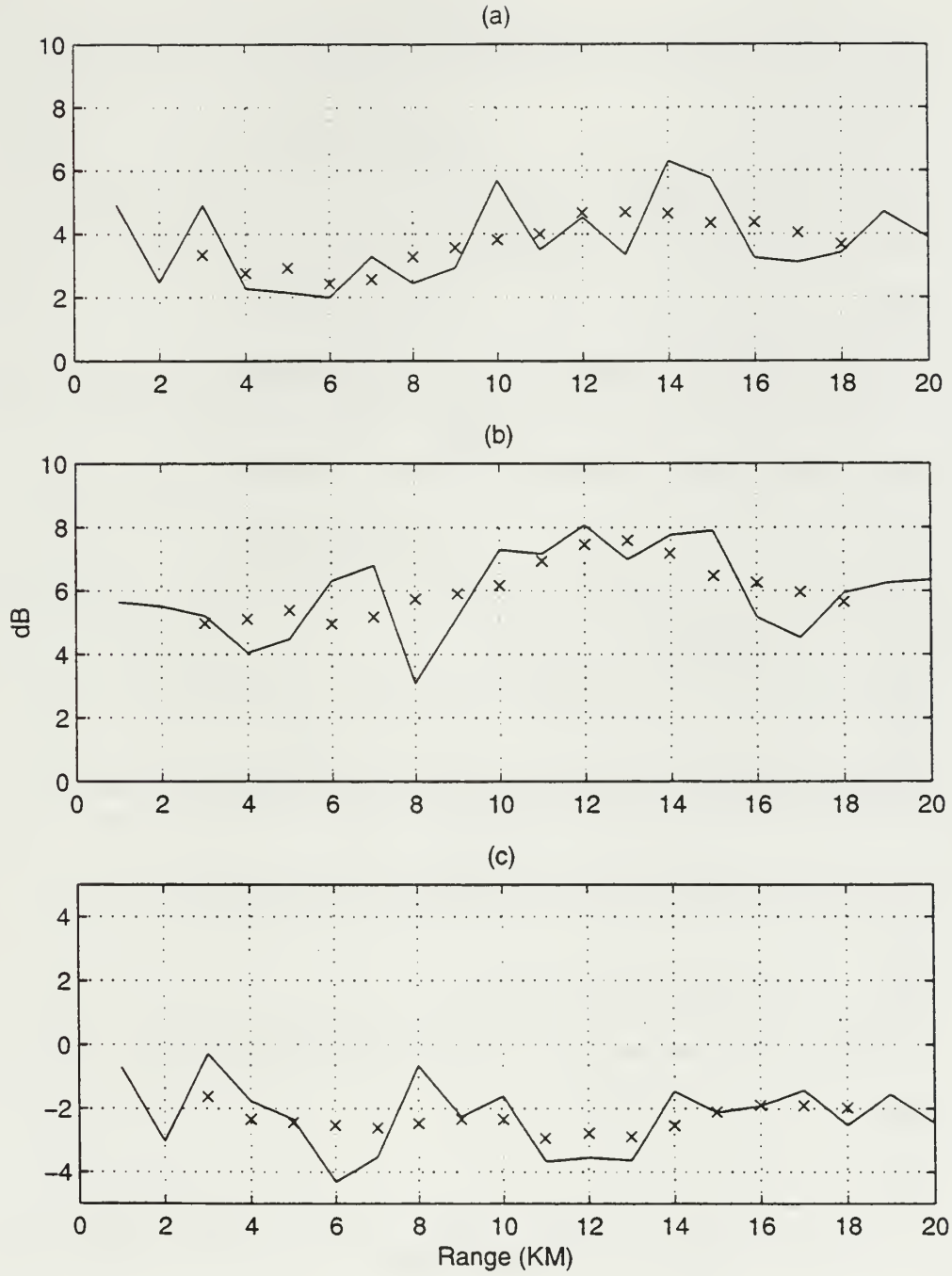

Figure 72. (a) MML plotted for run 1 with a 20 millisecond CW pulse. (b) MML plotted for run 1 with a 20 millisecond LFM pulse. (c) The MML difference between a CW pulse and a LFM pulse. (Graph properties as before)

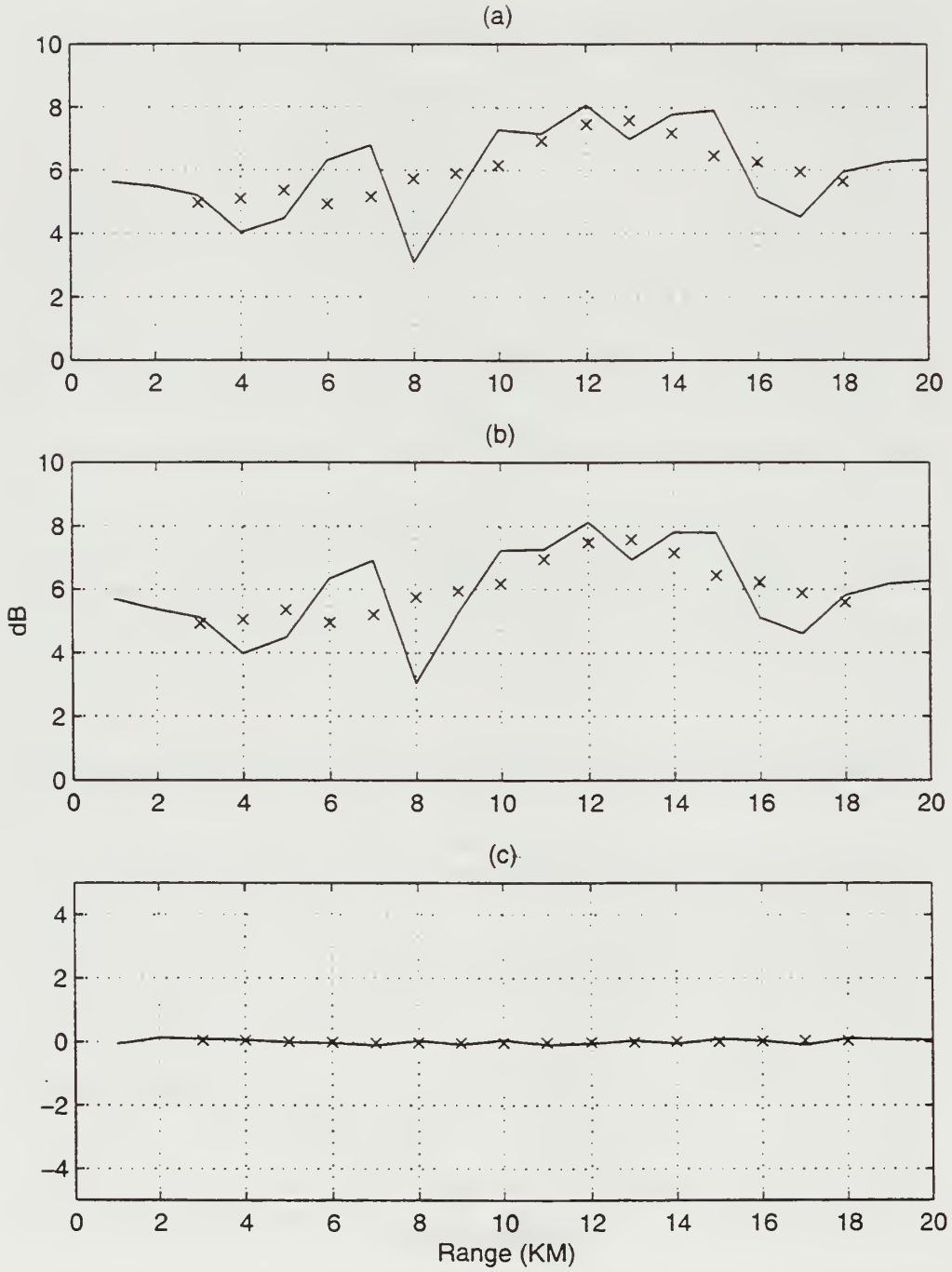

Figure 73. (a) MML plotted for run 1 with a 20 millisecond LFM pulse. (b) MML plotted for run 1 with a 20 millisecond HFM pulse. (c) The MML difference between an HFM pulse and a LFM pulse. (Graph properties as before)

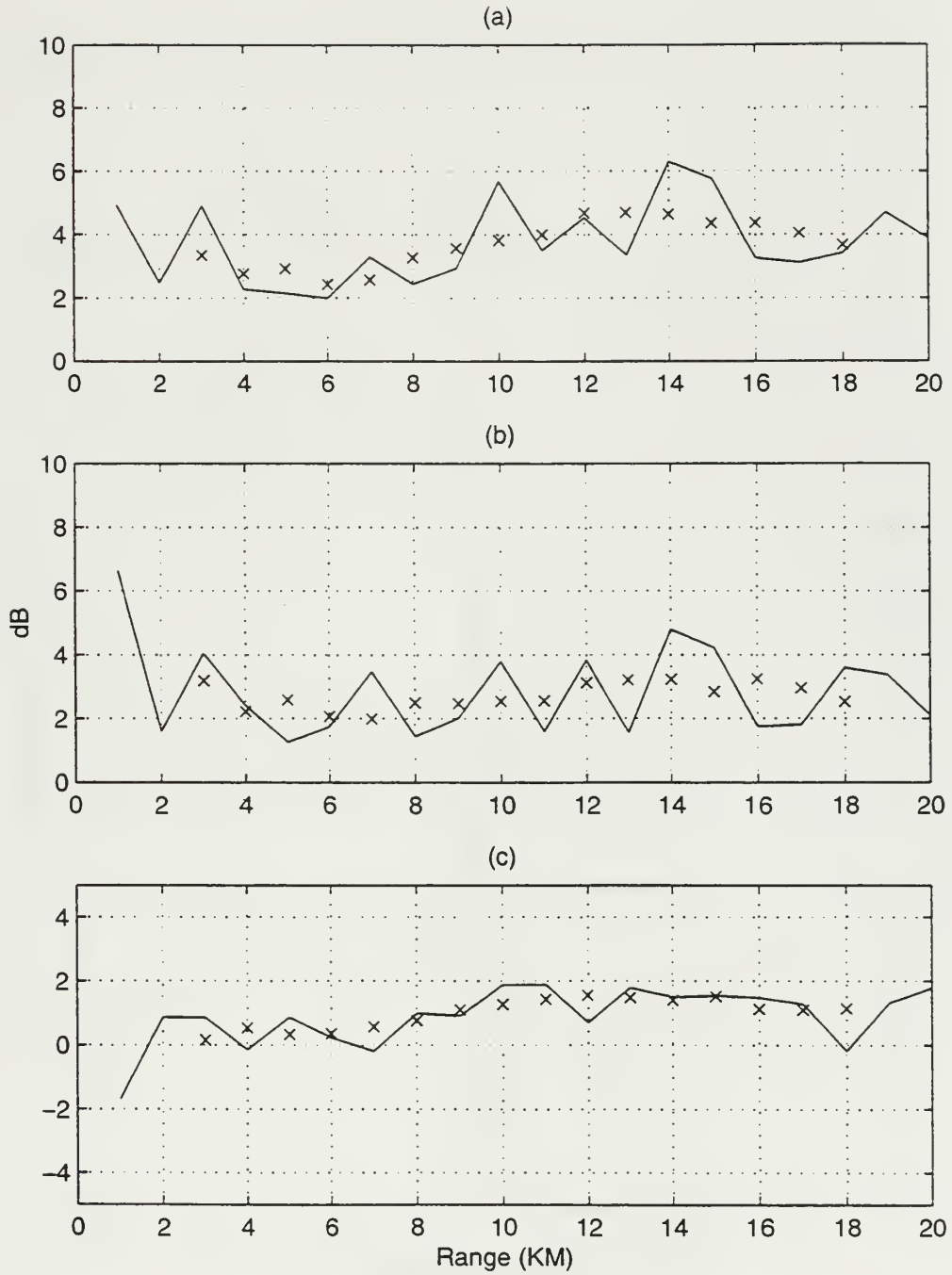

Figure 74. (a) MML plotted for run 1 with a 20 millisecond CW pulse. (b) MML plotted for run 1 with a 40 millisecond CW pulse. (c) The MML difference between a 20 and 40 millisecond CW pulse. (Graph properties as before)

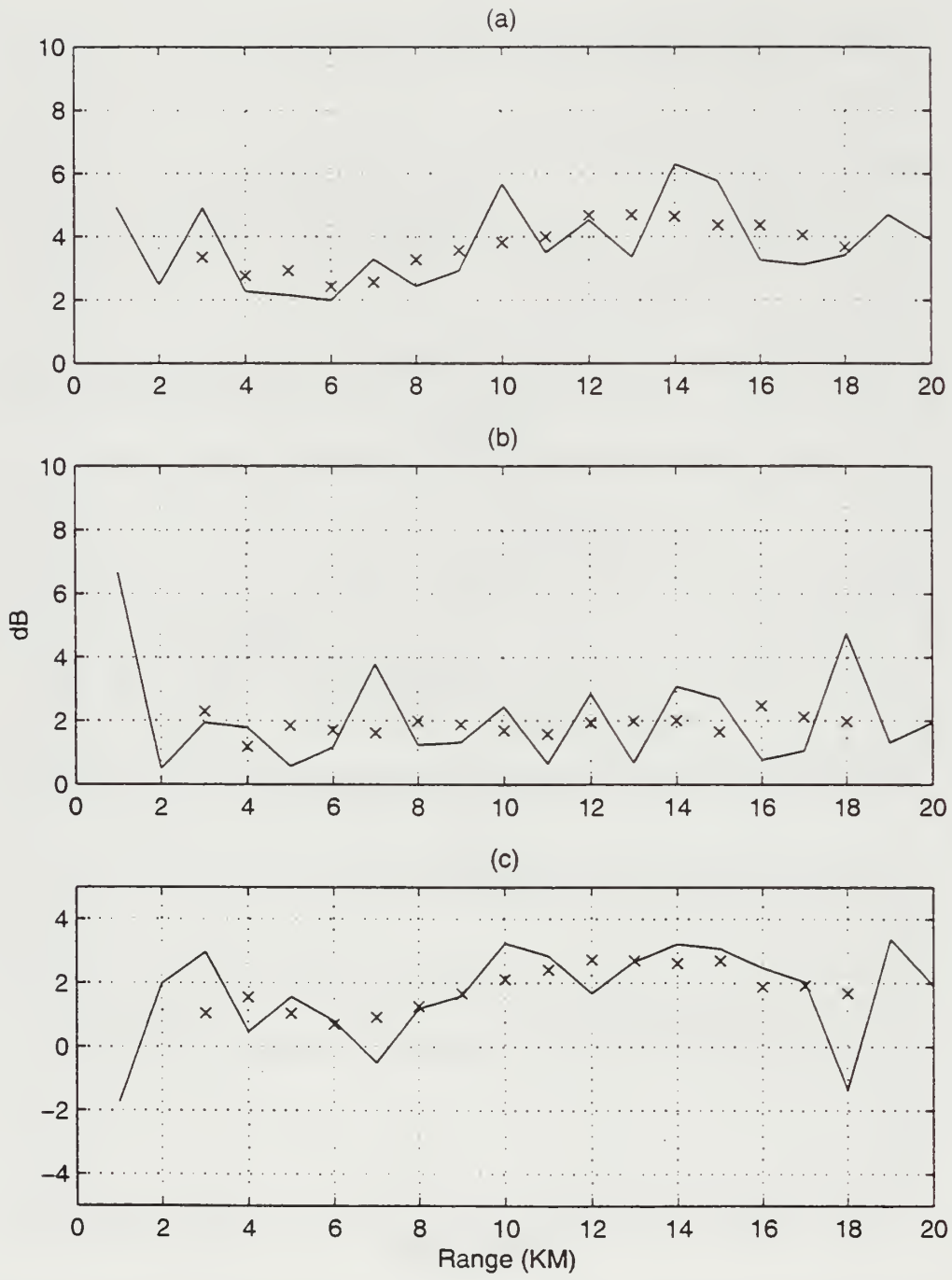

Figure 75. (a) MML plotted for run 1 with a 20 millisecond CW pulse. (b) MML plotted for run 1 with a 100 millisecond CW pulse. (c) The MML difference between a 20 and 100 millisecond CW pulse and a boxcar pulse. (Graph properties as before)

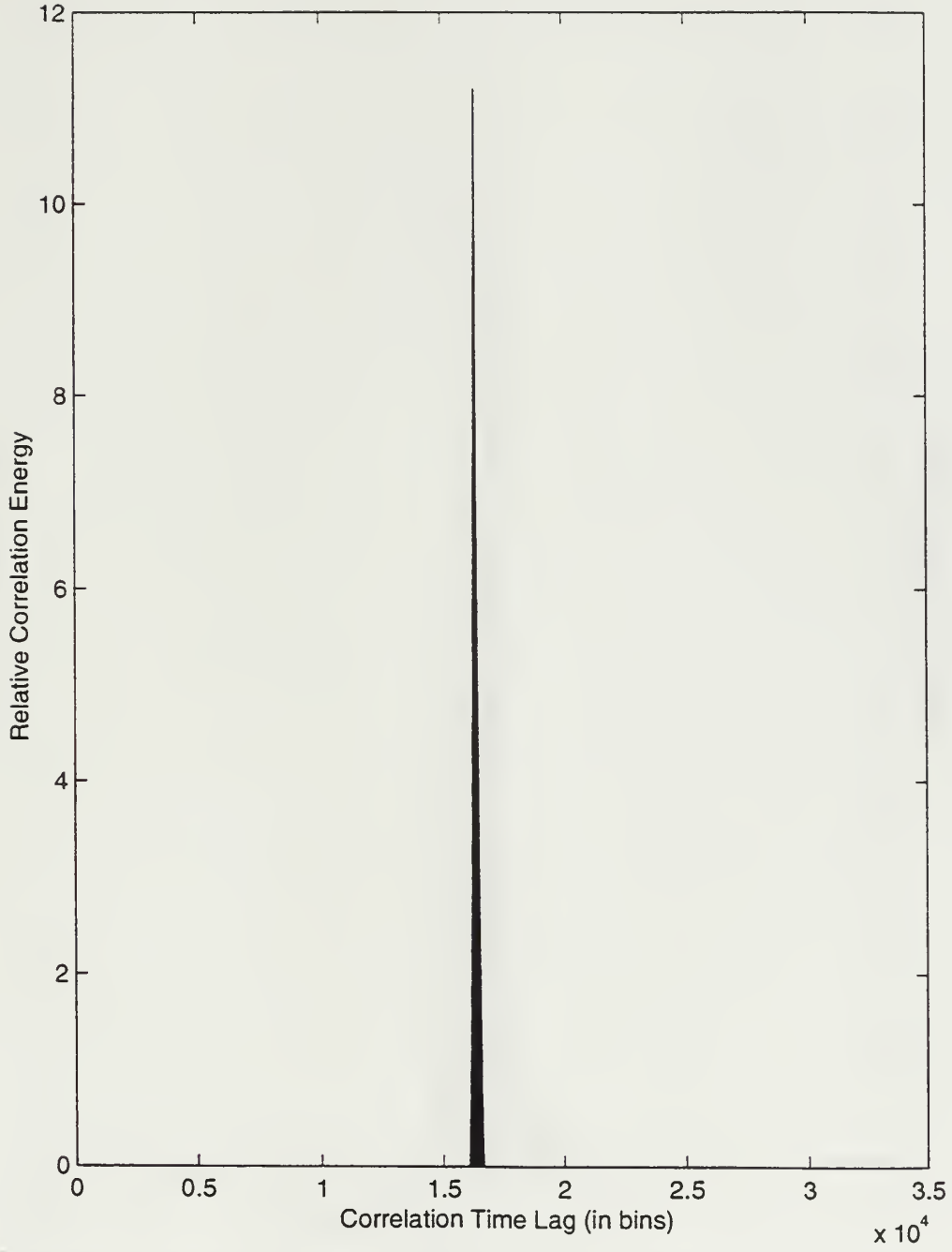

Figure 76. The graph represents the correlation between the 20 millisecond CW pulse and the compressed output pulse for run 1. The compressed pulse is a replica of the input pulse with the same amount of energy as the stretched pulse.

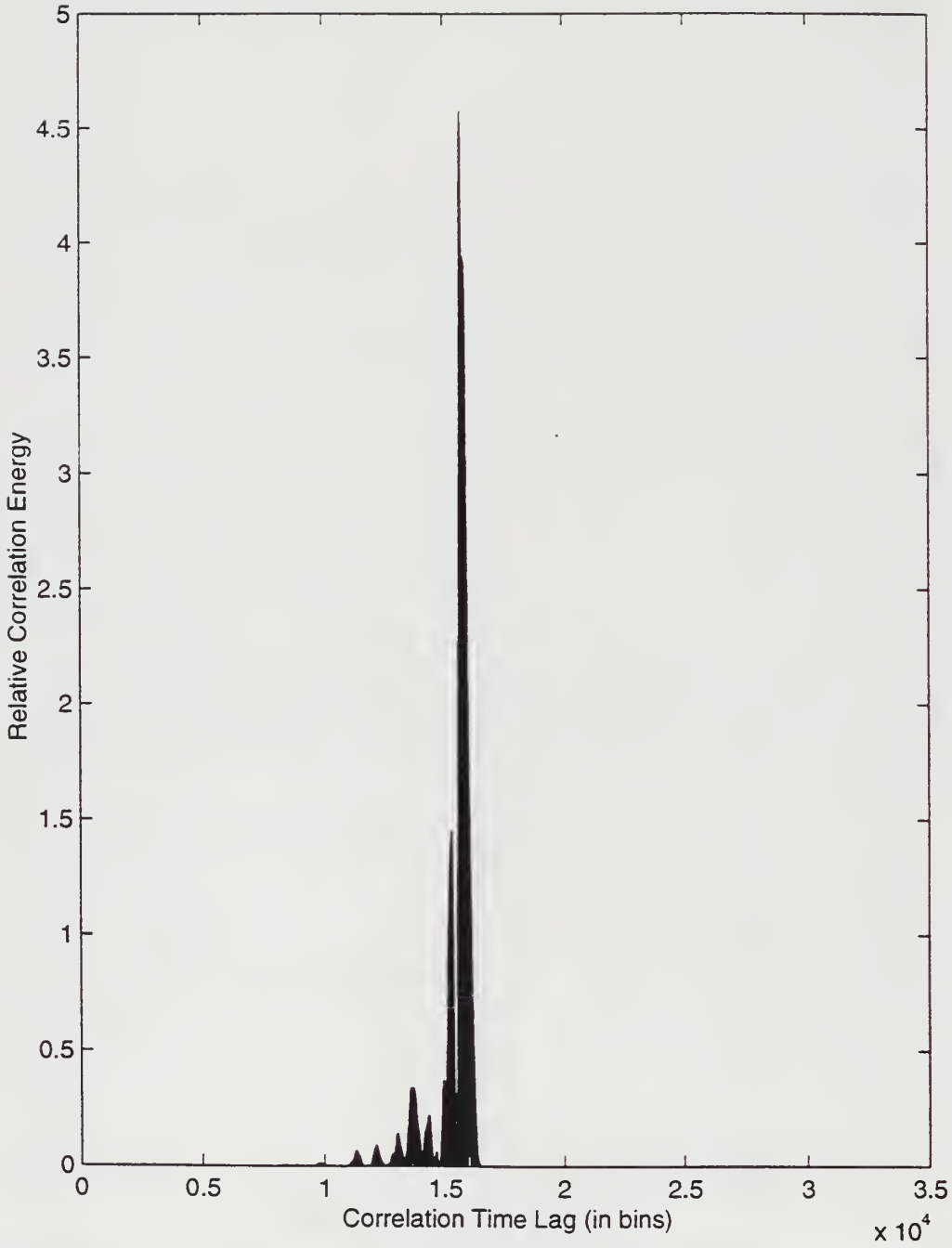

Figure 77. The graph represents the correlation between the 20 millisecond CW pulse and the stretched output pulse for run 1. The stretched pulse is the output of the OTF and input 20 millisecond CW pulse.

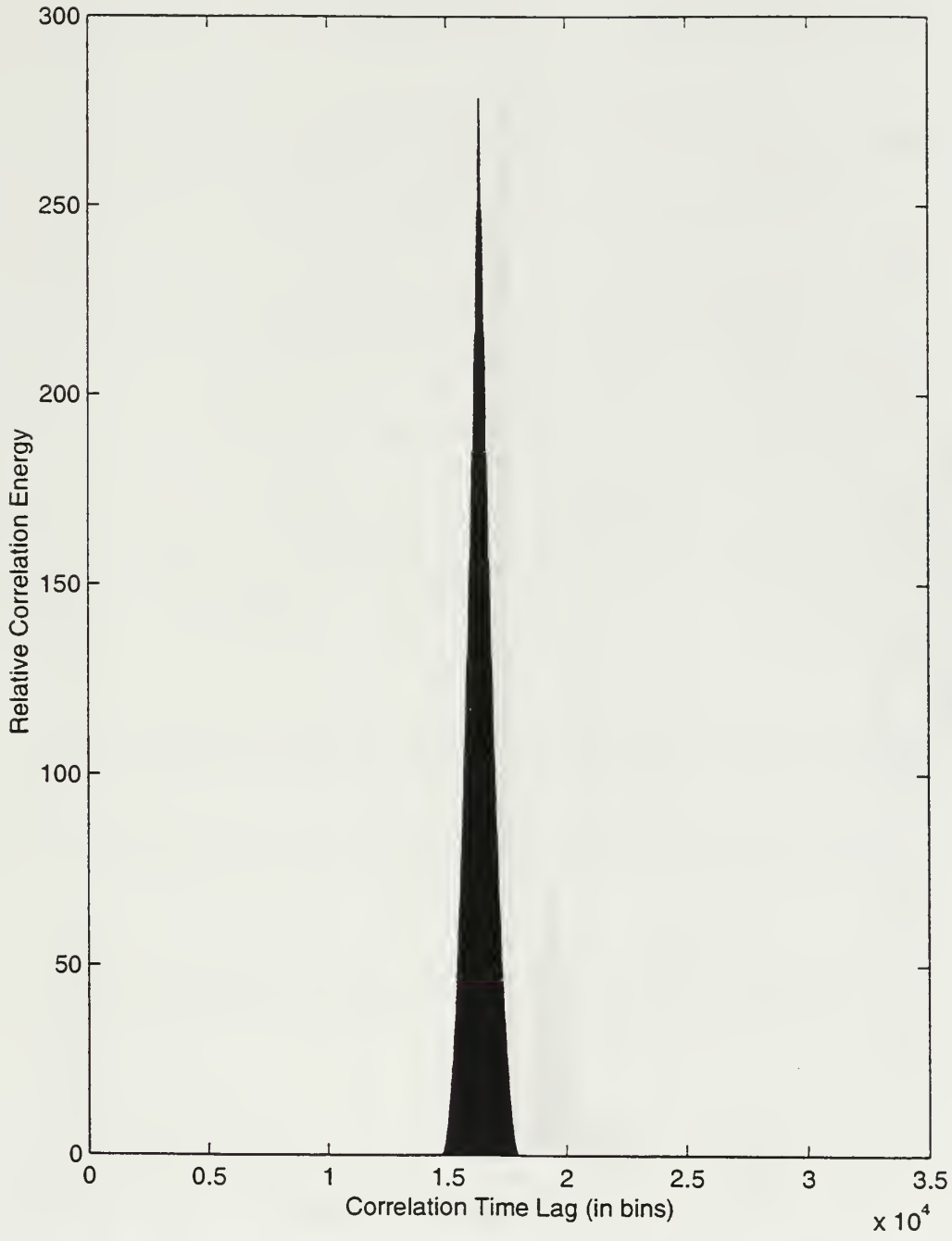

Figure 78. The graph represents the correlation between the 100 millisecond CW pulse and the compressed output pulse for run 1. The compressed pulse is a replica of the input pulse with the same amount of energy as the stretched pulse.

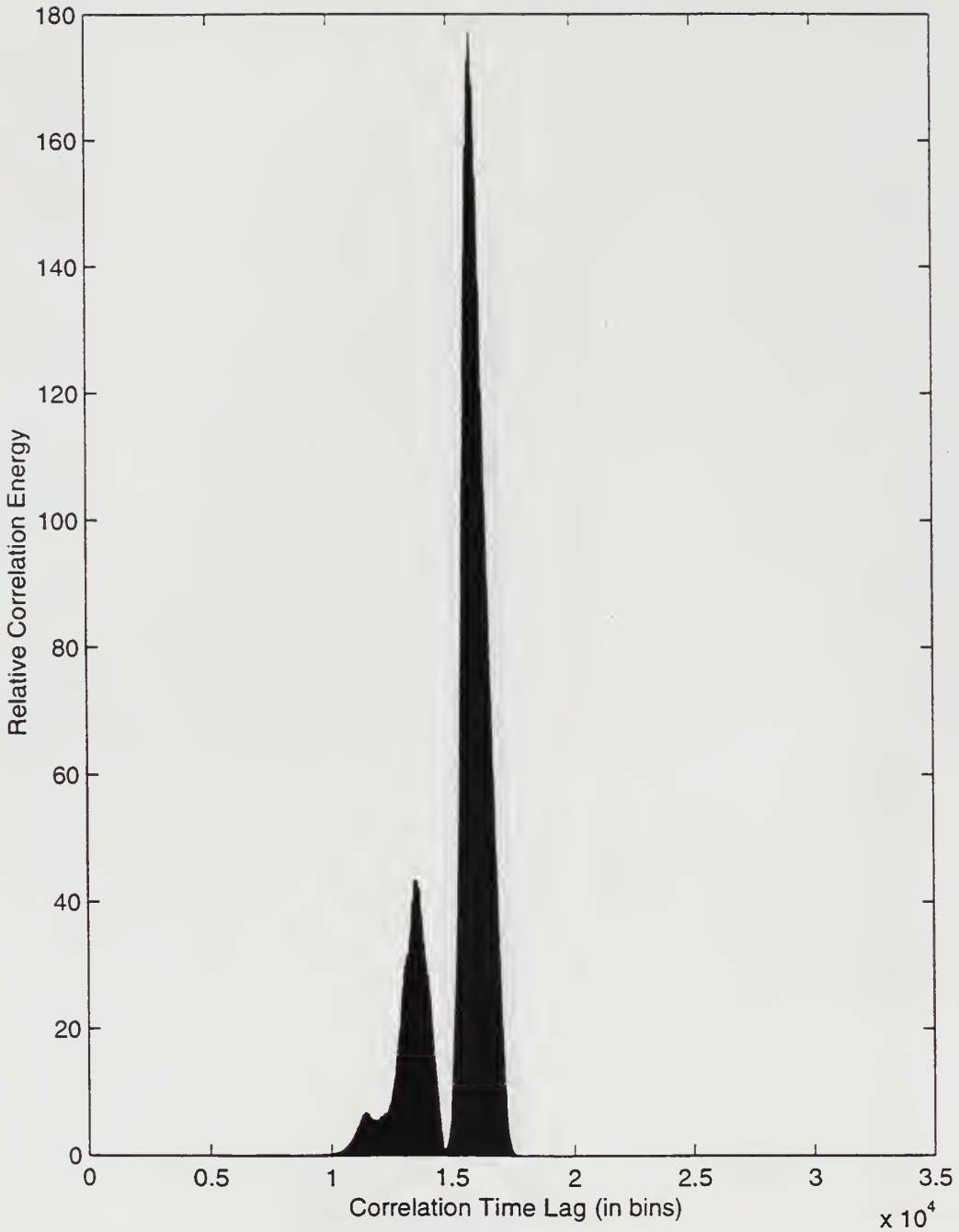

Figure 79. The graph represents the correlation between the 100 millisecond CW pulse and the stretched output pulse for run 1. The stretched pulse is the output of the OTF and input 100 millisecond CW pulse.

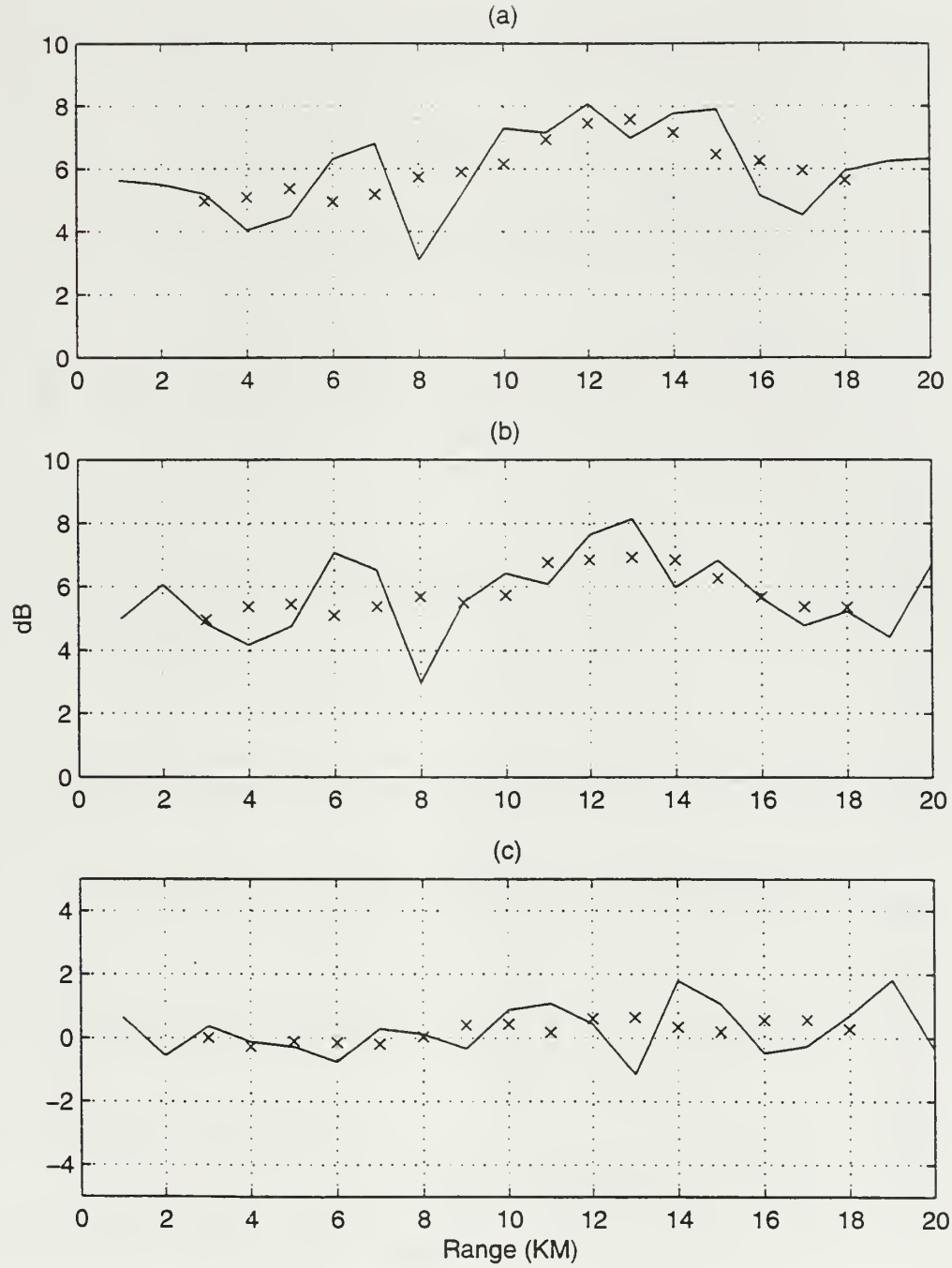

Figure 80. (a) MML plotted for run 1 with a 20 millisecond LFM pulse. (b) MML plotted for run 1 with a 40 millisecond LFM pulse. (c) The MML difference between a 20 and 40 millisecond LFM pulse. (Graph properties as before)

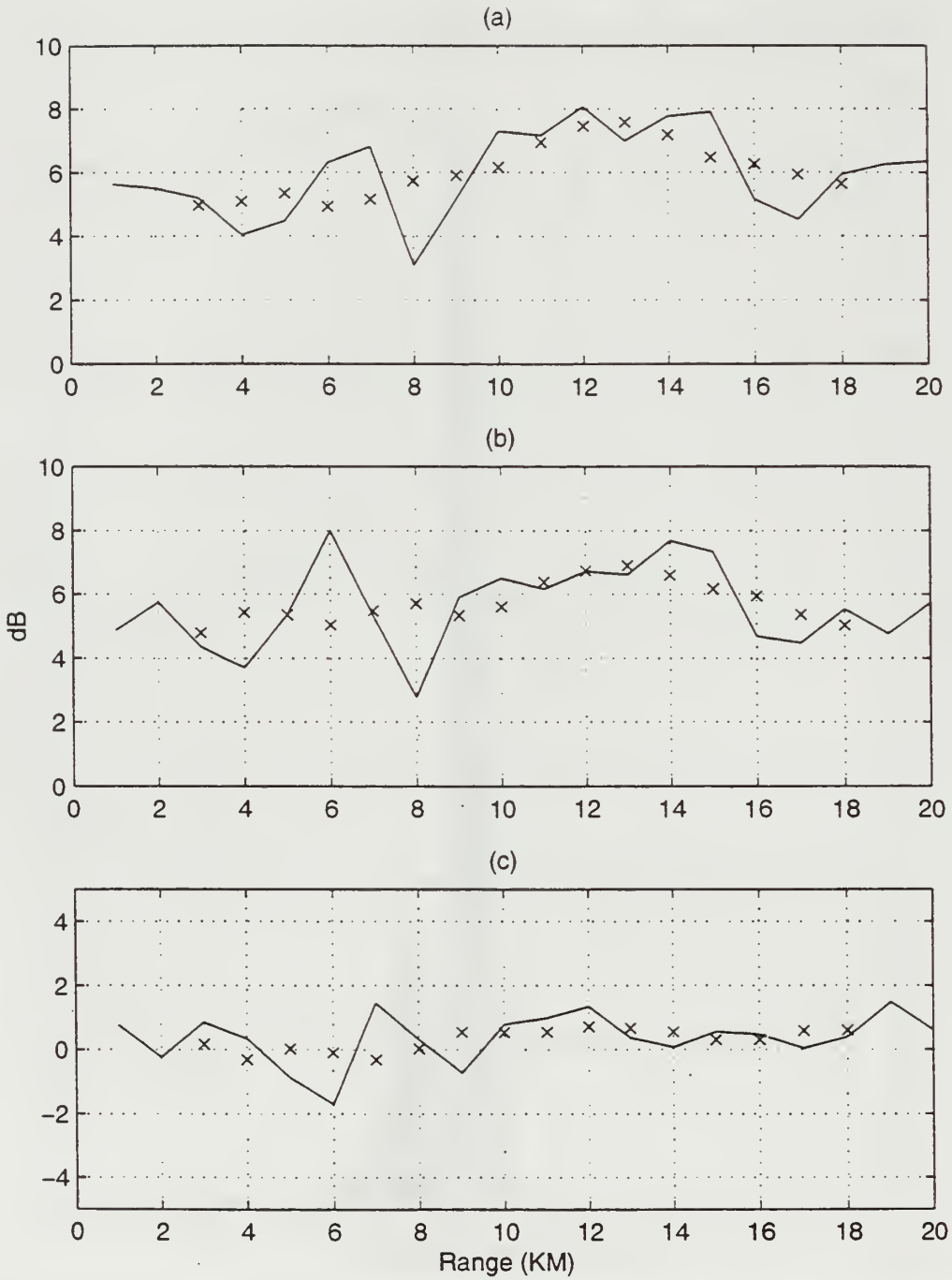

Figure 81. (a) ESL plotted for run 1 with a 20 millisecond LFM pulse. (b) ESL plotted for run 1 with a 100 millisecond LFM pulse. (c) The ESL difference between a 20 and 100 millisecond LFM pulse. (Graph properties as before)

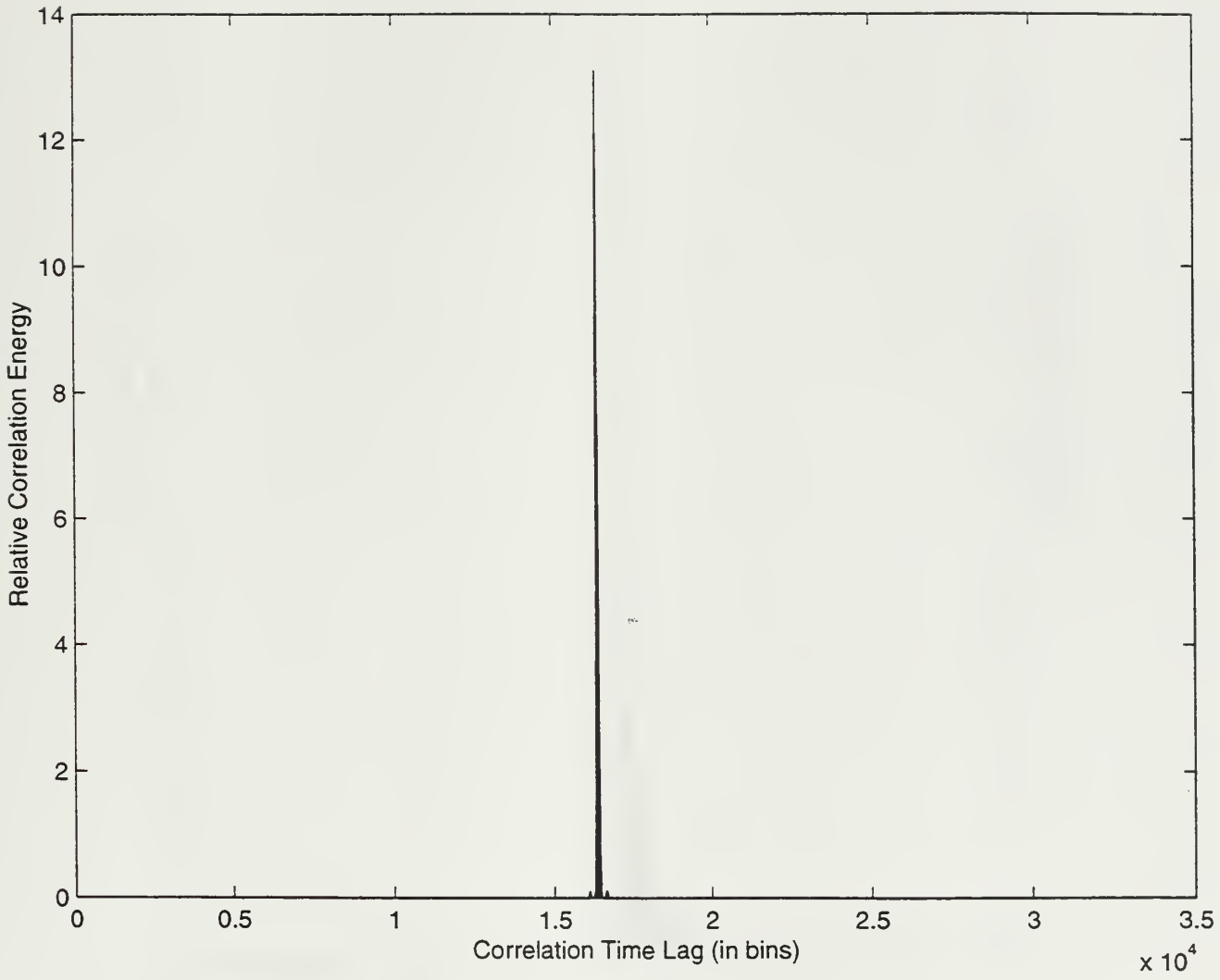

Figure 82. The graph represents the correlation between the 20 millisecond LFM pulse and the compressed output pulse for run 1. The compressed pulse is a replica of the input pulse with the same amount of energy as the stretched pulse.

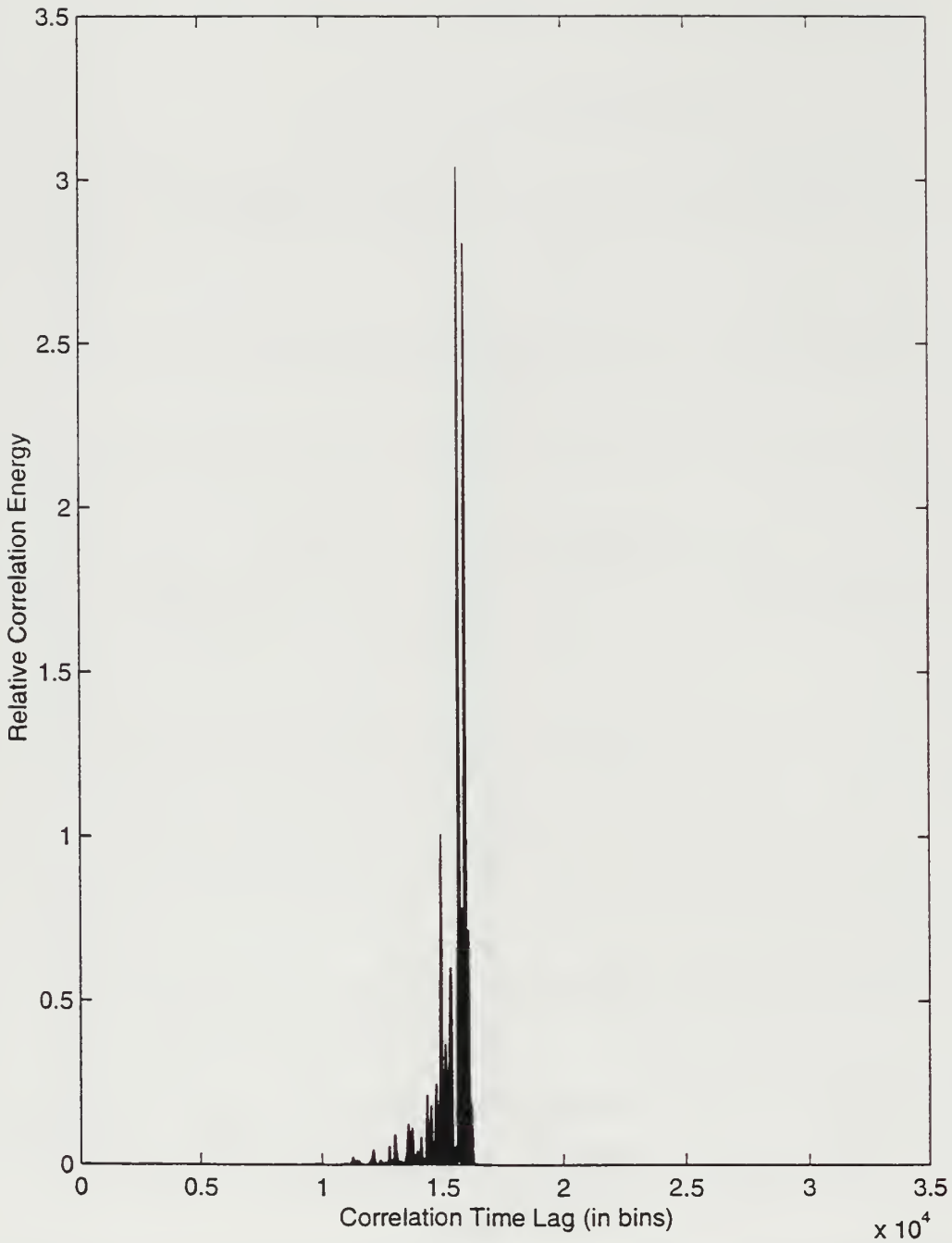

Figure 83. The graph represents the correlation between the 20 millisecond LFM pulse and the stretched output pulse for run 1. The stretched pulse is the output of the OTF and input 20 millisecond LFM pulse.

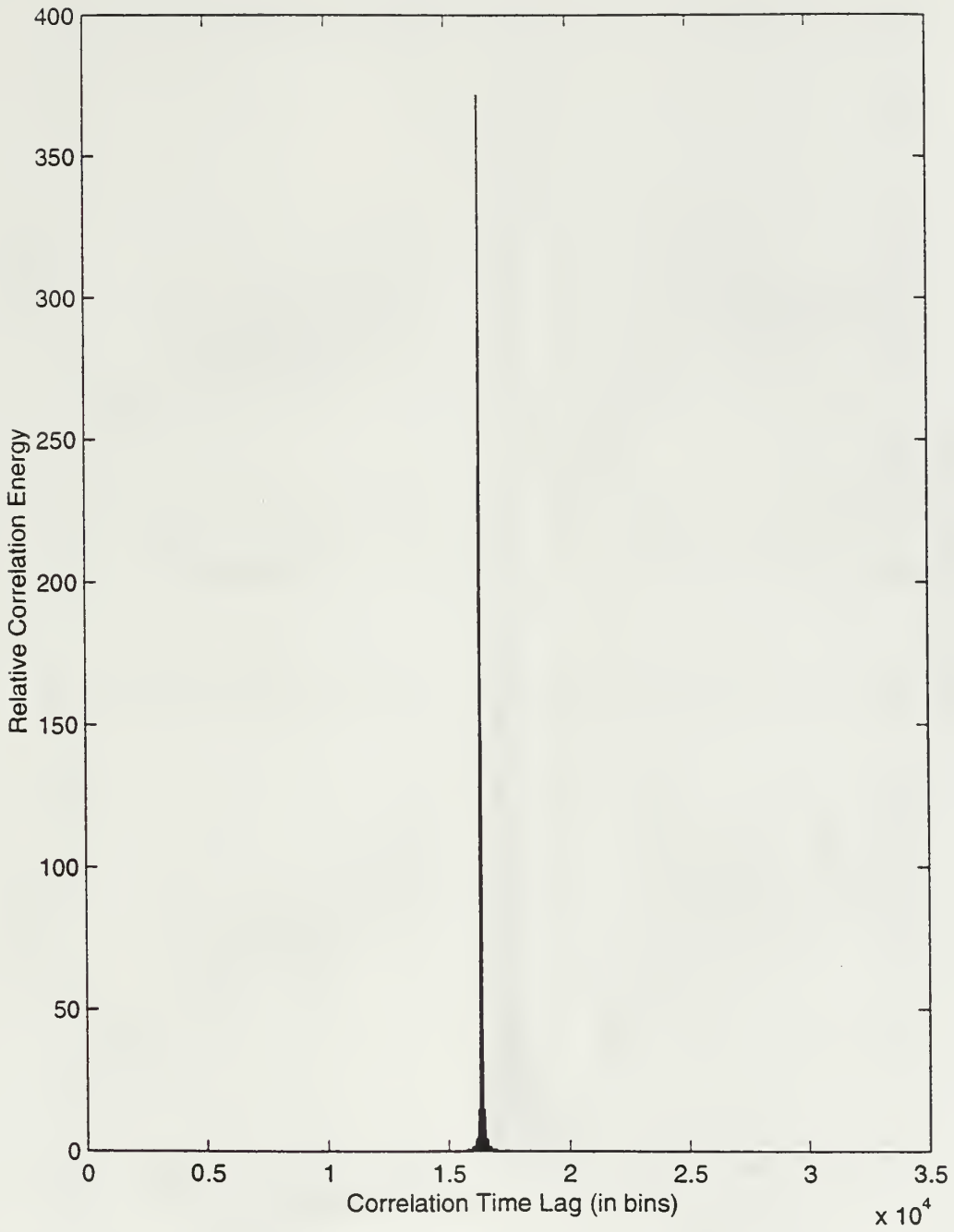

Figure 84. The graph represents the correlation between the 100 millisecond LFM pulse and the compressed output pulse for run 1. The compressed pulse is a replica of the input pulse with the same amount of energy as the stretched pulse.

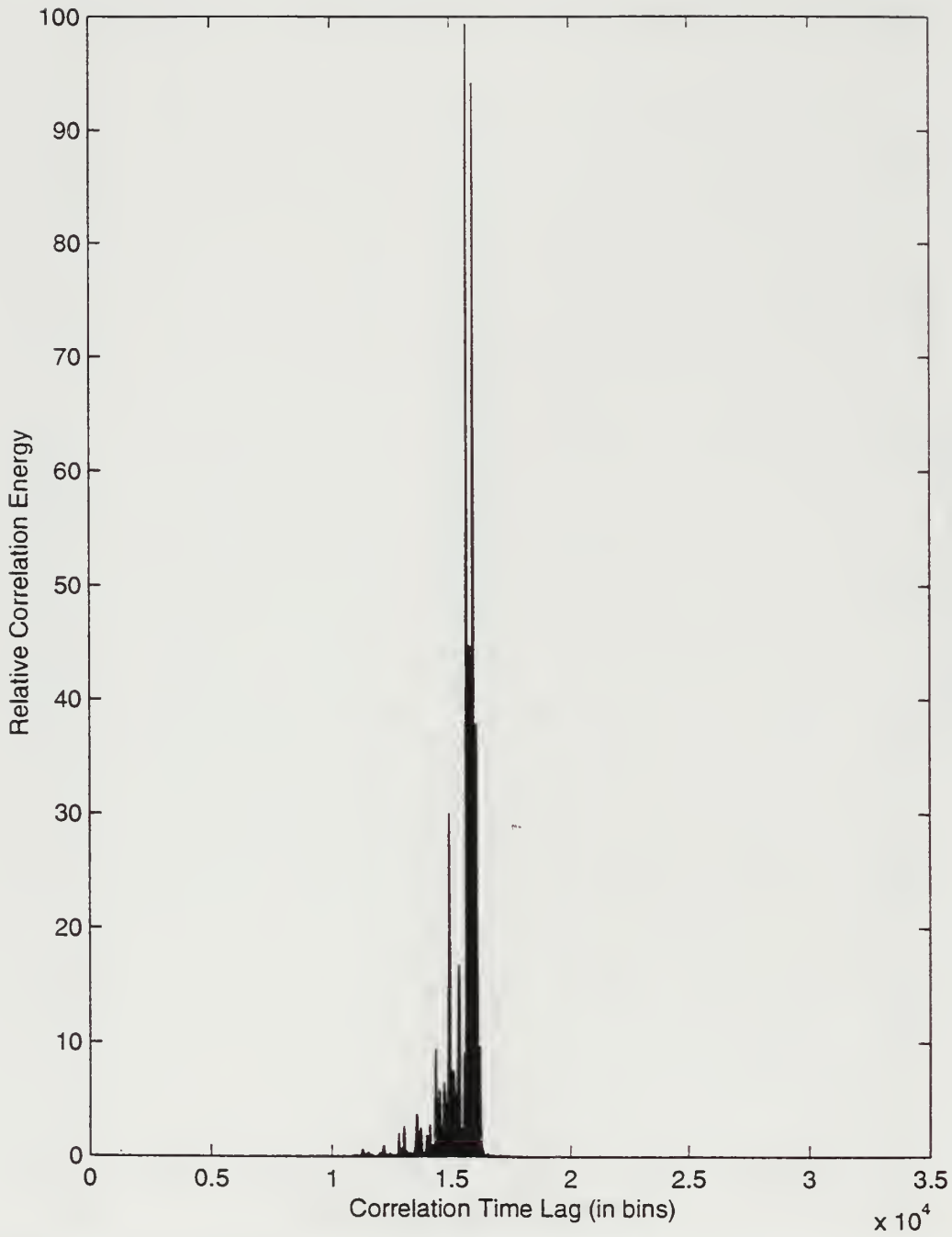

Figure 85. The graph represents the correlation between the 100 millisecond LFM pulse and the stretched output pulse for run 1. The stretched pulse is the output of the OTF and input 100 millisecond LFM pulse.

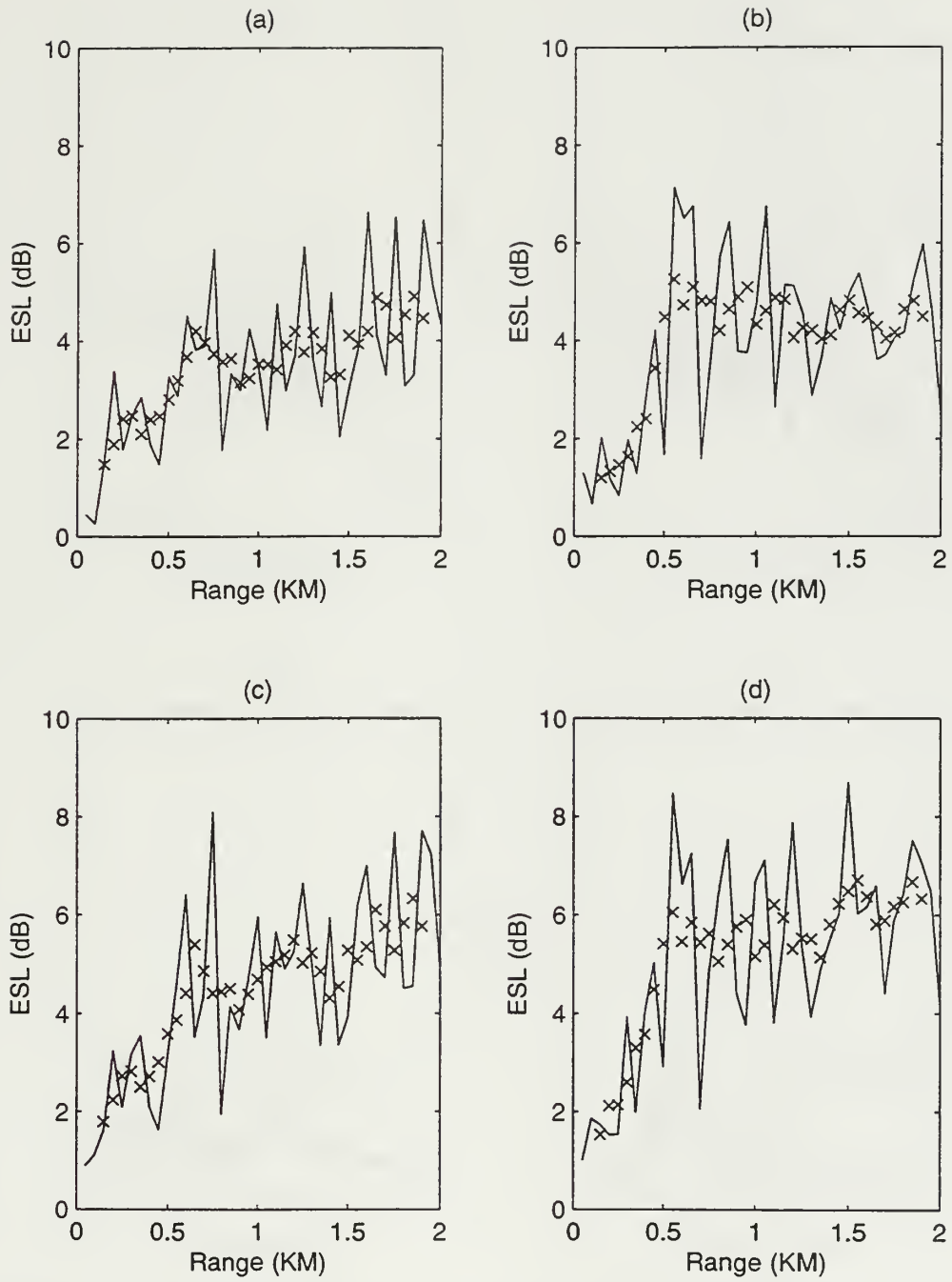

Figure 86. The graphs represent the ESL over the first 2000 m of a run. (a) run 1 with target depth at 11 m (b) run 1 with target depth at 22 m (c) run 11 with target depth at 11 m (d) run 11 with target depth at 22 m

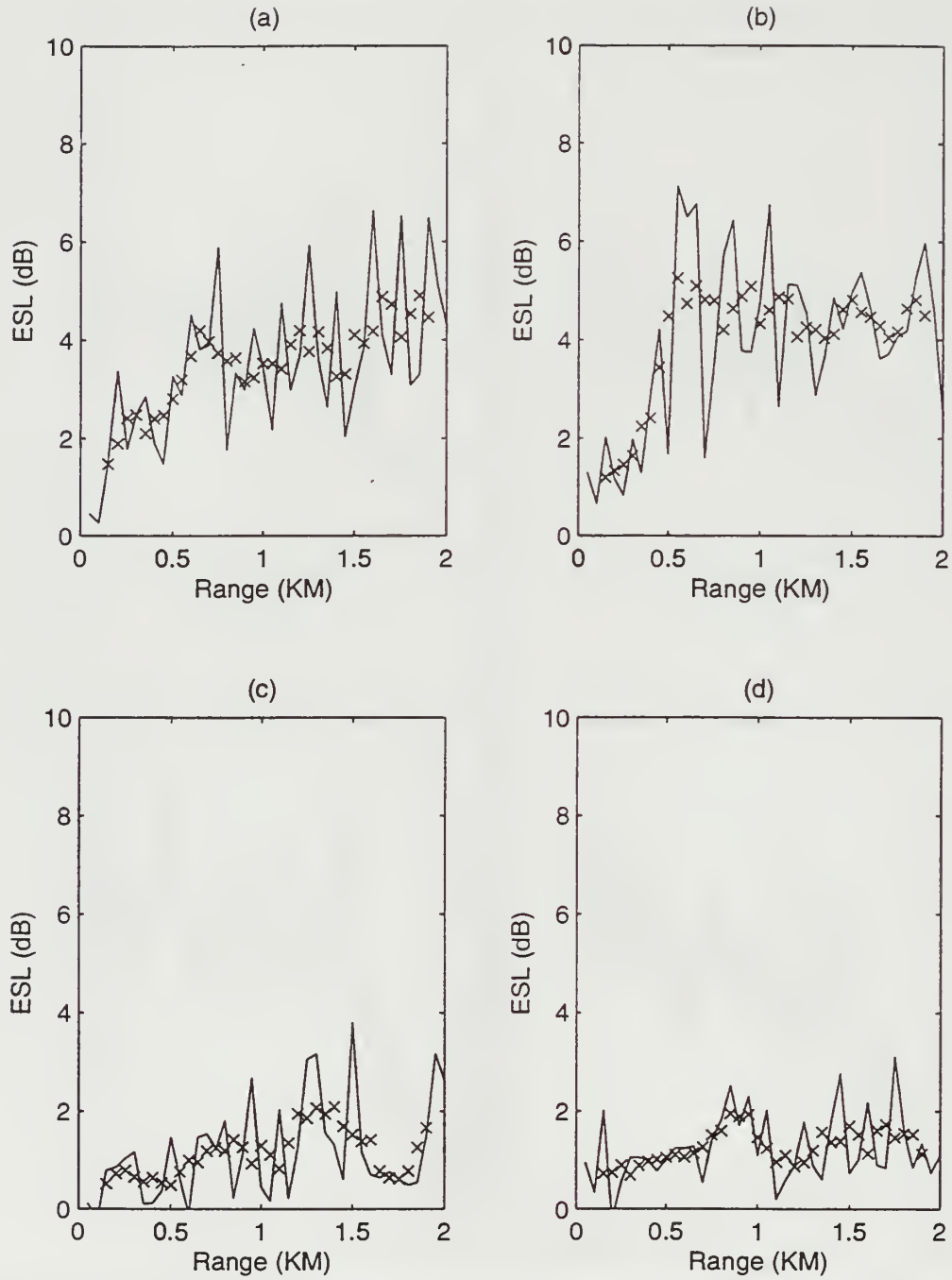

Figure 87. The graphs represent the ESL over the first 2000 m of a run. (a) run 3 with target depth at 11 m (b) run 3 with target depth at 22 m (c) run 10 with target depth at 11 m (d) run 10 with target depth at 22 m

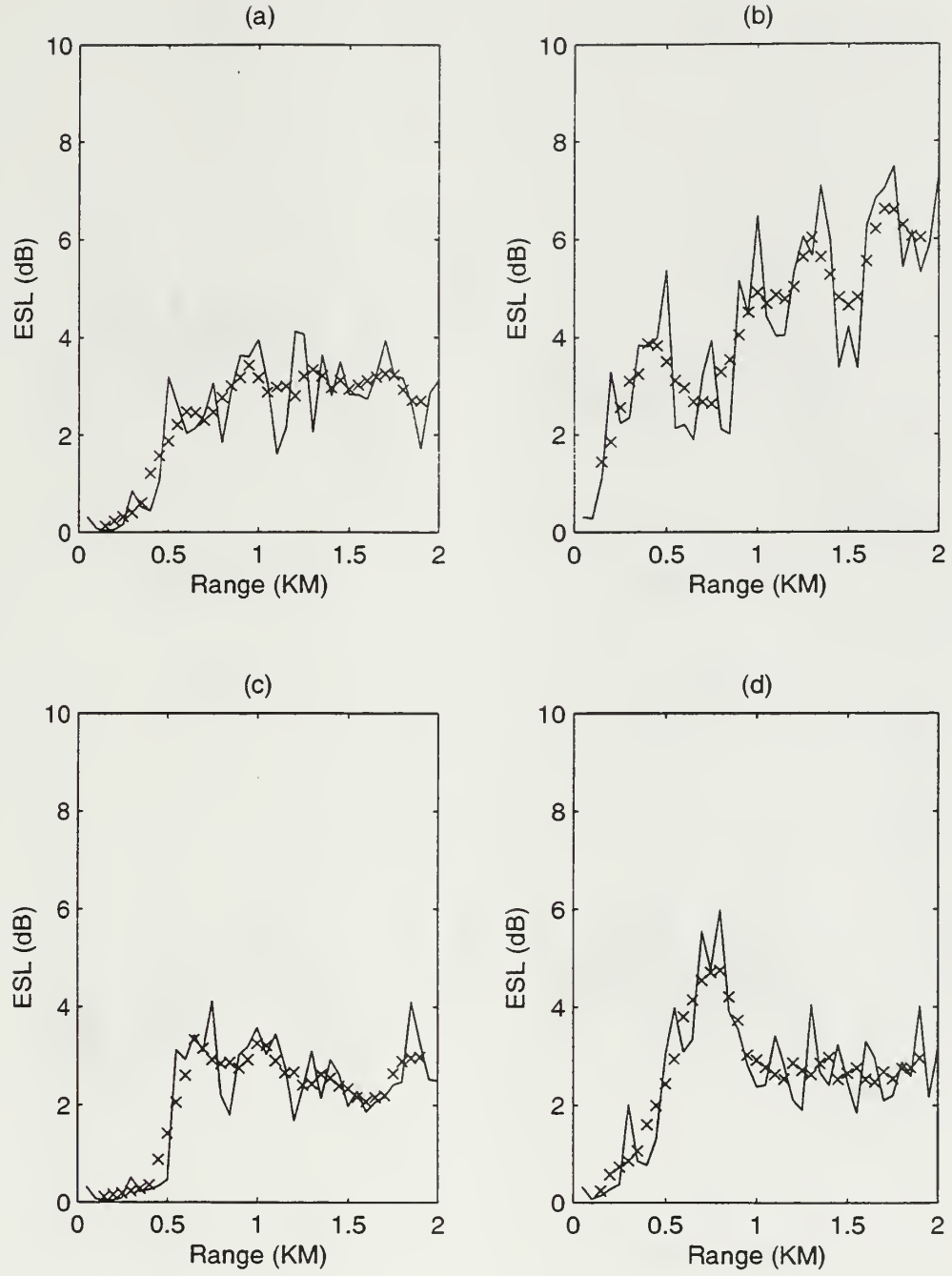

Figure 88. The graphs represent the ESL over the first 2000 m of a run. (a) run 22 with target depth at 11 m (b) run 24 with target depth at 22 m (c) run 26 with target depth at 11 m (d) run 21 with target depth at 22 m

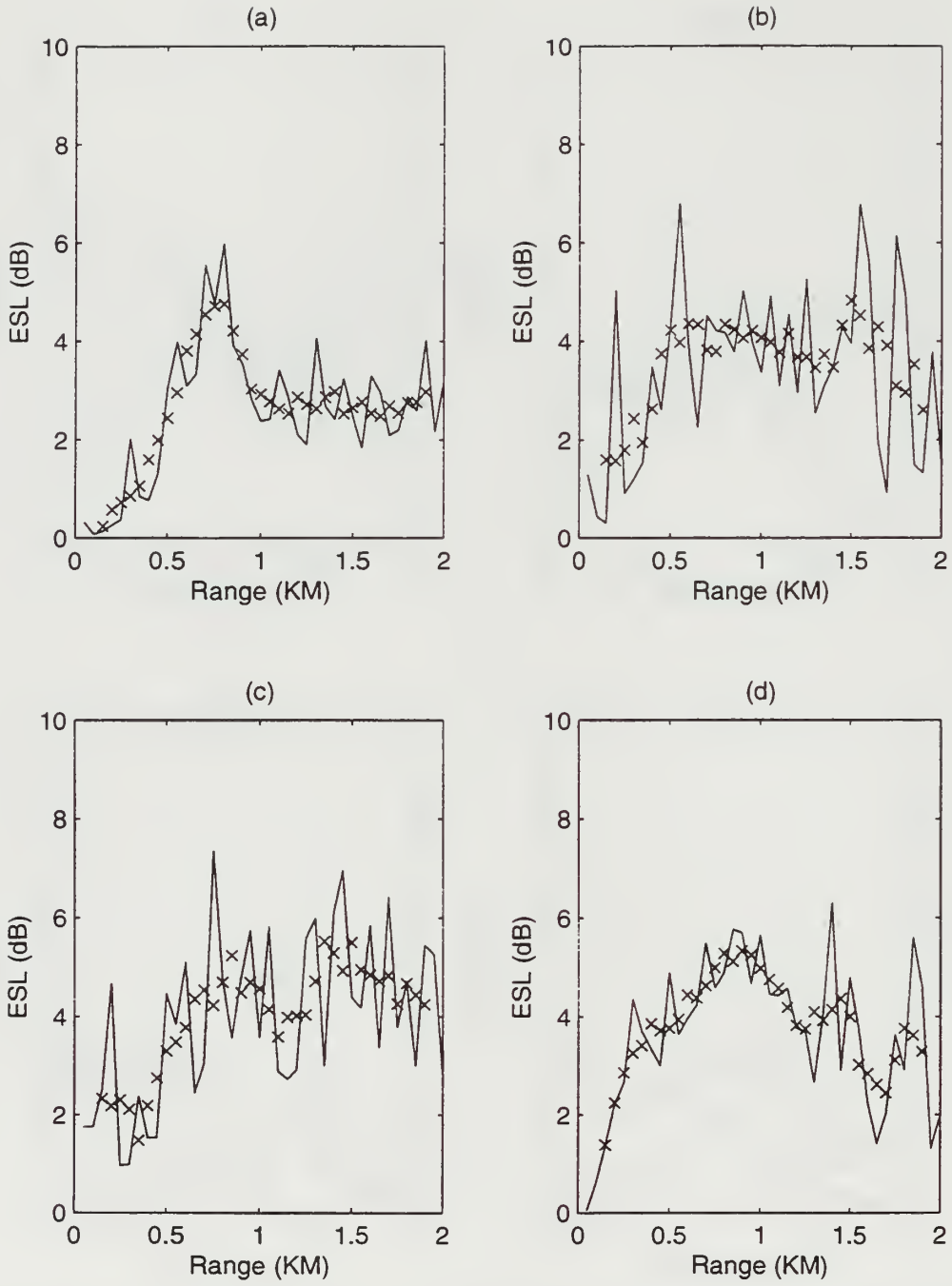

Figure 89. The graphs represent the ESL over the first 2000 m of a run. (a) run 21 with target depth at 11 m (b) run 21 with target depth at 33 m (c) run 30 with source depth at 55 m and target depth at 11 m (d) run 1 with source depth at 55 m and target depth at 55 m

APPENDIX B
MATLAB FEPE_SYN PROCESSING CODE

```
function[trunkx,avey,derx,dery]=avemove(x,y)
%function[trunkx,avey,derx,dery]=avemove(x,y)
%
%This function will take a row/column matrix of values and create a
%moving average.
%It will also trunkate the values that the average will be plotted against.
%
%input:          x,y
%                x - The values that y is plotted against. (ie range)
%                y - The values to be averaged. (ie TL)
%                (Y input should be in column form.)
%
%output:         trunkx, avey, derx, dery
%                trunkx - The x values trunkated so they can be used in plotting y
%                avey - The average values of y
%                derx - The x values trunkated so they can be used in plotting dery
%                dery - The approximate 1st derivative of avey  Calculated by taking
%                the rise of avey over the run of x
%
%Created: January 19, 1998
%         Peter Smith
%
%Last Modified: January 30, 1998
%        Peter Smith
%
%currently using a 5 point average. (ie a 2 point trunkation at each end.)

%trunkating x
lengthx = max(size(x));
trunk = x(3:(lengthx - 2));
dertrunk = x(3:(lengthx - 3));
lengthder = max(size(dertrunk));

%averageing y
lengthy = max(size(y));

y1=[y 0 0 0 0];
y2=[0 y 0 0 0];
y3=[0 0 y 0 0];
y4=[0 0 0 y 0];
y5=[0 0 0 0 y];

yaveraged = (y1 + y2 + y3 + y4 + y5)/5;
```

```
yfinal = yaveraged(5:lengthy);

%derivative of y
trunk1 = [trunk 0];
trunk2 = [0 trunk];
trunkdiff = trunk1 - trunk2;

yf1 = [yfinal 0];
yf2 = [0 yfinal];
ydiff = yf1 - yf2;

yderivative = (ydiff(1:lengthder))./(trunkdiff(1:lengthder));

%output
trunkx = trunk;
avey = yfinal;
derx = dertrunk;
dery = yderivative;
```

```
function[bartlettfreq]= bart(freqmin, freqmax, fftlength, dfreq)
%function[bartlettfreq]= bart(freqmin, freqmax, fftlength, dfreq)
%
%
%inputs:        freqmin,freqmax, fftlength,dfreq
%               freqmin - The lower frequency of the pulse.
%               freqmax - The upper frequency of the pulse.
%               fftlength - The number of points used in esl programs.
%               dfreq - The step in frequency for each fft point step.
%
%outputs:       bartlettfreq
%       The output is the frequency domain bartlett pulse.
%
%This function uses the signal processing bartlett function.

imin = floor(freqmin/dfreq) + 1;
imax = floor(freqmax/dfreq) + 1;
numberpoints = imax - (imin -1);

pulseshape = bartlett(numberpoints);

freqs = [zeros(imin-1,1);pulseshape;zeros((fftlength-imax),1)];

%output
bartlettfreq = freqs;
```

```
function[blackmanfreq]= black(freqmin, freqmax, fftlength, dfreq)
%function[blackmanfreq]= black(freqmin, freqmax, fftlength, dfreq)
%
%
%inputs:        freqmin,freqmax, fftlength,dfreq
%               freqmin - the lower frequency of the pulse
%               freqmax - the upper frequency of the pulse
%               fftlength -the number of points used in esl programs.
%               dfreq - the step in frequency for each fft point step
%
%outputs:       blackmanfreq
%               The output is the frequency domain blackman pulse.
%
%This function uses the signal processing blackman function.

imin = floor(freqmin/dfreq) + 1;
imax = floor(freqmax/dfreq) + 1;
numberpoints = imax - (imin-1);

pulseshape = blackman(numberpoints);

freqs = [zeros(imin-1,1);pulseshape;zeros((fftlength-imax),1)];

%output
blackmanfreq = freqs;
```

```
function[boxcarfreq]= boxpulse(freqmin, freqmax, fftlength, dfreq)
%function[boxcarfreq]= boxpulse(freqmin, freqmax, fftlength, dfreq)
%
%
%inputs:        freqmin,freqmax, fftlength,dfreq
%               freqmin - the lower frequency of the pulse
%               freqmax -the upper frequency of the pulse
%               fftlength -the number of points used in esl programs.
%               dfreq    -the step in frequency for each fft point step
%
%outputs:       boxcarfreq
%               The output is the frequency domain boxcar pulse.
%
%This function uses the signal processing boxcar function.
%
%Created: 15 December 1997
%Peter Smith
%Last Modified: 20 January 1998
%Peter Smith


imin = floor(freqmin/dfreq) + 1;
imax = floor(freqmax/dfreq) + 1;
numberpoints = imax - (imin -1);

pulseshape = boxcar(numberpoints);

freqs = [zeros(imin-1,1);pulseshape;zeros((fftlength-imax),1)];

%output
boxcarfreq = freqs;
```

```
function[correlation] = correlate(timeA, timeB);
%function[correlation] = correlate(timeA, timeB);
%
%This function correlates the two time series. If timeB has more than one
%time series stored in the matrix.  Thel function will divide them up and
%correlate each column with timeA;
%
%input:         timeA, timeB
%               timeA - the first time series
%               timeB - the second time series(can be a matirx of time series)
%
%output:        correlation
%               correlation - a matrix of the correlation between the time series
%
%
%uses: matlab function xcorr
%
%Created: 14 February 1998
%       Peter Smith
%Last Modified: 12 March 1998
%       Peter Smith
n=size(timeB);

%Brute force method
corrmatrix = [];
for o = 1:n(2);
  o
  outputcorr = xcorr(timeA,timeB(:,o));
  corrmatrix = [corrmatrix outputcorr];
end% for o

%fft method
%numbercorr = min(size(timeB));
%C = [];
%A = fft(timeA);
%B = fft(timeB);
%for  n = 1:numbercorr
%  output = B(:,n) .* A;
%  C = [C output];
%end % for n
%corrmatrix = real(ifft(C));
%output
correlation = corrmatrix;
```

```
function[timescale,cwwave]= cw(pulsetimemin,pulsetimemax,fcent,fftlength,tmax)
%function[timescale,cwwave]= cw(pulsetimemin,pulsetimemax,fcent,fftlength,tmax)
%file cw.m
%
%This function will return a time domain cw pulse back to
%the calling program
%
%Input:      pulsetimemin,pulsetimemax,fcent,fftlength,tmax
%           pulsetimemin - the begining time of the pulse
%           pulsetimemax - the ending time of the pulse (pulselength =max-min)
%           fcent - the frequency of the pulse
%           fftlength - the number of points used in the calling program for
%           vectors in both time and frequency domain.
%           tmax - the length of time related to last bin in time domain
%           timestep = tmax/fftlength
%
%
%Output:     cwwave
%           cwwave- time domain pulse of fcent freqency
%
%Created: 14 December 1997
%Peter E. Smith
%
%Last Modified: 14 December 1997
%Peter E. Smith

%getting the frequency in radians
w0 =  2*pi*fcent;
%geting indicies for time
n=   1:fftlength;
dt = tmax/fftlength;
N= n*dt;
binmin = floor(pulsetimemin/dt);
binmax = floor(pulsetimemax/dt);
binlength = binmax - binmin;

%setting up wave to specifications from calling program
cosinewave = cos(w0*N);
wave=[zeros(1,binmin) cosinewave(1:binlength) ...
    zeros(1,fftlength - binmax)];
%output
cwwave = wave;
timescale = N;
```

```
function[energyspreading,  TL]= esl(rawsig, output, fftsize)
%function[energyspreading,  TL]= esl(rawsig, output, fftsize)
%
%This matlab function will use the information that should be called from the
%previous functions extotr, and (a signal generation function).
%The signal generation function will have been chosen previously by the user.
%This program compares the timeseries of the raw signal with the time series
%of the signal at the ranges selected.
%
%Input:        rawsig, otr (freq domain zero padded)
%              rawsig -the sonar pulse being evaluated
%              otr -    the ocean transfer function
%              fftsize - the size of fft being used
%
%Output:       energyspreading, TL
%              energyspreading- the peak processing signal loss due to energy spreading
%              TL - the dB result of r
%
%Created: December 12, 1998
%       Peter Smith
%
%Last Modified: Jan 17, 1998
%       Peter Smith
rawtime = real(ifft(rawsig,fftsize));
timeseries = real(ifft(output,fftsize));

%energy of the original outgoing waveform
Eout = sum(rawtime.^2);
%energy of the stretched pulse
Enospread = sum(timeseries.^2);
r = (Enospread/Eout);
%max amplitude of the waveform which has no time spreading
Anospread = r*max(rawtime.^2);

%max amplitude of the waveform which has time spreading
Aspread = max(timeseries.^2);
%ESL
absorbTL=-10*log10(r);
sl =-10*log10(Aspread./Anospread);

%output
energyspreading = sl;
TL = absorbTL;
```

```
function[esl2way,tl2] = plotesl(rawfreqs,otr,otr2,freqmin,freqmax,fftsize,...
                        dfreq,ranges,scatter)
%function[esl2way, tl2] = plotesl(rawfreqs,otr,otr2,freqmin,freqmax,fftsize,...
%                         dfreq,ranges,scatter)
%
%This program will calculate and plot the esl and tl for the OTR's chosen.
%
%Input:         rawfreqs,otr,freqmin,freqmax,fftsize,dfreq,ranges,pulsetitle,fileid
%               rawfreqs - the initial signal pulse.
%               otr - the ocean transfer function.
%               otr2 - the second ocean transfer function.
%               freqmin - the minimum frequency of otr.
%               freqmax - the maximum frequency of otr.
%               fftsize - the size of fft being used in program.
%               dfreq - the frequency step size in fft.
%               ranges - the ranges relating to the coulumns of the otr for plotting.
%
%Output:        esl2way
%               esl2way - the esl from a 2 way distance.
%
%Uses:          outputsig.m, esl.m, avemove.m
%
%Created: 14 February 1998
%Peter Smith
%Last Modified:14 February 1998
%Peter Smith
%

a=min(size(otr));
b=min(size(otr2));



output2way =  outputsig2(rawfreqs,otr(:,a),otr2(:,b),scatter,...
                 freqmin,freqmax,fftsize,dfreq);
[esl2, transmissionloss2] = esl(rawfreqs, output2way, fftsize);

%output
esl2way = esl2;
tl2 = transmissionloss2;
```

```
%File: ESLcommand.m
%
%This program will run the esl programs in the 'esl' process file.
%This program is primarily the GUI interface between user
%and the binary output file of fepesyn.
%
%The files that are to be used in this program need to have the
%same field name.  The extenstion name also needs to be
%of the type .in .out .log.  When requested, only enter field argument.
%(ie onslow1.in onslow1.out onslow1.log)
%
%The programs that are called by the ESLcommand.m do not need to be in the
%same dirrectory as the command file.  The path should be amended using the
%path command to route matlab to the dirrectory that you hold the commands.
%
%Created: 15 December 1997
%       Peter Smith
%
%Last Modfied: 11 March 1998
%       Peter Smith
%
%Requires Matlab 5.0 for reading input file
%(Matlab 5 will ignore comment lines with a %.  For earlier models the
% comment line needs to be erased.)
%
%Required from matlab: Matlab 5, Signal Processing Toolbox

%the path for reading the m-files-specific for each situation
%computer center path
%path(path,'/tmp_mnt/h/alioth_u0/pesmith/thesis.dir/fepesyn.dir/process.dir')
%
%oc computer path
%path(path,'puffin_u4/pesmith/thesis.dir/fepesyn.dir/process.dir')

%setup
clear


startflag = 1;
%Overall loop
endflag = 1;
while endflag == 1
```

```
%Beginning input
if startflag  == 1
  start = menu('What do you want to do?','Run Input');
end %if == 1


if startflag == 2
  start = menu('What do you want to do?','Run Input','Binary Input',...
            'MAT-input');
end %if == 2

if startflag ==3
  start = menu('What do you want to do?','Run Input','Binary Input',...
            'MAT-input',...
            'Save OTR-MAT File','Create Pulse','Plot OTR');
end %if ==3

if startflag >= 4
  start = menu('What do you want to do?','Run Input','Binary Input',...
            'MAT-input',...
            'Save OTR-MAT File','Create Pulse','Plot OTR',...
            'Plot Pulse Shape(FREQ)','Plot Pulse Shape(TIME)',...
           'Plot Output Signal(FREQ)','Plot Output Signal(TIME)',...
            'Plot ESL',...
            'Movie Output Signal(TIME)','Change Input Variables',...
            'Plot MML','2-way ESL','2-way MML','Compare ESL',...
            'Compare MML');
  end %if >= 4

startflag = startflag +1;

if start == 1 %Run Input
  readinput;
end % start 1

if start == 2 %Binary Input
  %hardwire
  %rangerequest = 1000:1000:20000;

  rangerequest = input('What ranges do you want? >> ');
  depthrequest = input('What depth do you want? >> ');
  otr = extotr(fileid, recordsize, freqmin, freqmax, dfreq,...
            ranges, rangerequest, depths, depthrequest);
```

```
end % start

if start == 3 %Matlab MAT file input
  otr = otrmatload;
end % start 3

if start == 4 %Save OTR-MAT File
  otrmatsave(otr);
end % start 4

if start == 5 %Create Pulse
  pulseshape;
end % start 5

if start == 6 %Plot OTR
  rangerequest = input('What ranges do you want? >> ');
  otrplot(otr,ranges, freqmin, freqmax, fftsize, dfreq, rangerequest,...
        fileid);
end % start 6

if start == 7 %Plot Pulse Shape (FREQ)
   plotpulsef(rawfreqs,freqmin,freqmax,dfreq,fftsize,pulsetitle);
end % start 7

if start == 8 %Plot Pulse Shape (TIME)
   plotpulset(rawfreqs,dtime,fftsize,pulsetitle);
end % start 8

if start == 9 %Plot Output Signal (FREQ)
   rangerequest = input('What ranges do you want? >> ');
  outputplotf(rawfreqs, otr, ranges, freqmin, freqmax, fftsize,...
                dfreq, rangerequest, fileid);
end % start 9

if start == 10 %Plot Output Signal (TIME)
  rangerequest = input('What ranges do you want? >> ');
  outputplott(rawfreqs, otr,ranges, freqmin, freqmax, fftsize, dfreq,...
           dtime,rangerequest,fileid );
end % start 10

if start == 11 %Plot ESL
  plotesl(rawfreqs,otr,freqmin,freqmax,fftsize,dfreq,ranges,...
                pulsetitle,fileid)
```

```
 end % start 11

 if start == 12 %Movie Output Signal (TIE)
  rangerequest = input('What ranges do you want? >> ');
  outputmoviet(rawfreqs, otr,ranges, freqmin, freqmax, fftsize, dfreq,...
          dtime,rangerequest,fileid );
 end % start 12

 if start == 13 %Change Input Variables
  inputchange
 end % start 13

 if start == 14 %Plot MML
    plotmml(rawfreqs,otr,freqmin,freqmax,fftsize,dfreq,...
         ranges,pulsetitle,fileid);
 end % start 14

 if start == 15 %2-Way ESL
  otr2=otrmatload;
  scatter = [zeros(1,3399) ones(1,201) zeros(1,16384-3600)]';
  [esl2,tl2]=esl2way(rawfreqs,otr,otr2,freqmin,freqmax,fftsize,...
              dfreq,ranges,scatter)
 end % start 15

 if start == 16 %2-Way MML
  otr2=otrmatload;
  scatter = [zeros(1,3399) ones(1,201) zeros(1,16384-3600)]';
  [mml2,tl2]=mml2way(rawfreqs,otr,otr2,freqmin,freqmax,fftsize,...
              dfreq,ranges,scatter)
 end % start 16

 if start == 17 %Compare ESL
  otr2=otrmatload;
  plotcompesl(rawfreqs,otr,otr2,freqmin,freqmax,fftsize,dfreq,...
       ranges,pulsetitle,fileid,tmin,tmax);
 end % start 17
 if start == 18 %Compare MML
  otr2=otrmatload;
  plotcompmml(rawfreqs,otr,otr2,freqmin,freqmax,fftsize,dfreq,...
       ranges,pulsetitle,fileid,tmin,tmax);
 end % start 18
 endflag = menu('Do you want to continue?','Yes','No');
end % while end
```

```
flagfunction[otr] = extotr(filename,recordsize,freqmin,freqmax,...
                freqstep,ranges,rangerequest,...
                depths,depthrequest)
%function[otr] = extotr(filename,recordsize,freqmin,freqmax,...
%                 freqstep,ranges,rangerequest,...
%                    depths,depthrequest)
%
%This function extracts the ocean transfer function from binary file
%
%This output file used to be read by fortran program.
%The first record which is the input file can not be read by matlab
%unless the input file is strictly written in floating point varialbles
%Floating point and integer data in binary code can only be read seperately
%in matlab. I chose to ignore the first record and read only the
%ocean transfer function data.
%
%Input:      fileid,recordsize,numberranges,freqmin,freqmax,ranges,rangerequest
%           depths depthrequest
%           fileid - The output file that will be read
%           recordsize - The size of each record. It is used with the number
%           of ranges to determine the matlab skip factor for
%           reading data. The skip factor is in bytes and
%           not in the number of numbers skipped.
%           Record size is known as Isize in input file for fepesyn.
%           freqmin - The minimum frequency saved.
%           freqmax - The maximum frequency saved.
%           freqstep - The step size used in saving the program.
%           ranges - The ranges that were saved. (vector)
%           rangerequest - The ranges requested to be returned from the function.
%           These ranges must be equal to one of the ranges in the
%           ranges function or it will not be looked up.  This
%           might crash the program later.
%           depths - The depths that were saved. (vector)
%           rangerequest - The depths requested to be returned from the function.
%            These depths must be equal to one of the ranges in the
%            ranges function or it will not be looked up.
%
%
%Output:otr
%           The otr function can be either a sigle column vector or a matrix of
%           column vectors with each column being a different range.
%
%
```

```
%Created: 15 December 1997
%Peter E. Smith
%
%Last Modified: 21 January 1997
%Peter E. Smith



%setting up variables for later use
%dummy variables
Areal = [];
Aimag = [];
A = [];
B= [];
initialrecords = [];

%needed for number of loops per range request
numberfreqs = floor((freqmax - freqmin)/freqstep) +1;



%finding initial skip to the initial frequency record
numrequest = max(size(rangerequest));
initialdepth = find(depthrequest == depths) - 1;
for m = 1:numrequest;
  initialfound = find(rangerequest(m) == ranges);
  if isempty(initialfound)
    warning = ['The following range request index was not found in the ranges:']
    m
   end % if
  initialrecords = [initialrecords initialfound];
  clear initialfound
end %for m
initialskip = ((recordsize+1) *4)*ones(size(initialrecords)) ...
          + ((initialrecords -1 )*recordsize*4) + (initialdepth-1)*8;



%finding the matlab skip factor for reading multiple record files
%(see help fread)
numberranges = max(size(ranges));
skipfactor = ((recordsize - 1) * 4) +...
```

```
        (recordsize * (numberranges - 1) *4);

%open output file for reading only
fileid = [filename,'.out'];
fid = fopen(fileid,'r');


%read first record which contains input information from infile and skip
%to the first data point -(note first file size is  recordsize +3 ???)

%get real data
for o = 1:numrequest;
 %holding variables for loop
 o
 A=[];
 Areal=[];
 Aimag=[];

 %For troubleshooting-this should always equal the first input of the '.
 %'.in' file
 frewind(fid);
 first = fread(fid,1,'float32',initialskip(o));

 %get real data
 for p = 1:numberfreqs
  realpart = fread(fid,1,'float32',skipfactor);
  Areal = [Areal; realpart];
 end % for p

 frewind(fid);
 first = fread(fid,1,'float32',initialskip(o)+4);
 %get imaginary data
 for r = 1:numberfreqs
  imagpart = fread(fid,1,'float32',skipfactor);
  Aimag = [Aimag; imagpart];
 end % for r
 A=Areal +i* Aimag;
 B= [B A];
end %for o
%closing '.out' file
fclose(fid);
%output
otr = B;
```

```
function[hammingfreq]= ham(freqmin, freqmax, fftlength, dfreq)
%function[hammingfreq]= ham(freqmin, freqmax, fftlength, dfreq)
%
%
%inputs:        freqmin,freqmax, fftlength,dfreq
%               freqmin - the lower frequency of the pulse
%               freqmax -the upper frequency of the pulse
%               fftlength -the number of points used in esl programs.
%               dfreq - the step in frequency for each fft point step
%
%outputs:       hammingfreq
%               The output is the frequency domain hamming pulse.
%
%This function uses the signal processing hamming function.

imin = floor(freqmin/dfreq) + 1;
imax = floor(freqmax/dfreq) + 1;
numberpoints = imax - (imin -1);

pulseshape = hamming(numberpoints);

freqs = [zeros(imin-1,1);pulseshape;zeros((fftlength-imax),1)];

%output
hammingfreq = freqs;
```

```
function[hanningfreq]= han(freqmin, freqmax, fftlength, dfreq)
%function[hanningfreq]= han(freqmin, freqmax, fftlength, dfreq)
%

%
%inputs:        freqmin, freqmax, fftlength,dfreq
%               freqmin - the lower frequency of the pulse
%               freqmax - the upper frequency of the pulse
%               fftlength - the number of points used in esl programs.
%               dfreq    - the step in frequency for each fft point step
%
%outputs:       hanningfreq
%               The output is the frequency domain hanning pulse.
%
%This function uses the signal processing hanning function.

imin = floor(freqmin/dfreq) + 1;
imax = floor(freqmax/dfreq) + 1;
numberpoints = imax - (imin -1);

pulseshape = hanning(numberpoints);

freqs = [zeros(imin-1,1);pulseshape;zeros((fftlength-imax),1)];

%output
hanningfreq = freqs;
```

```
function[timescale, hfmwave] = hfm(pulsetimemin,pulsetimemax,...
                    fftsize,totaltime,freqmin,freqmax);
%function[timescale, hfmwave] = hfm(pulsetimemin,pulsetimemax,...
%                           fftsize,totaltime,freqmin,freqmax);
%
%This function creates a hyperbolic frequency modulated waveform
%
%Input:         pulsetimemin,pulsetimemax,centerfreq,fftsize,totaltime,freqmax
%               pulsetimemin - starting time for HFM pulse
%               pulsetimemax - ending time for HFM pulse
%               centerfreq -  the middle frequency of the sweep
%               fftsize - the size of the fft being used
%               totaltime - the sampling time being used
%               freqmax - the upper limit of the HFM pulse.
%
%Output =       timescale - the timescale for the hfm wave
%               hfmwave - the hfm wave in time domain
%
%
% Hyperbolic Frequency Modulated waveform
% Reference:  Drumheller, David M. "Uniform Spectral AMplitude Windowing
%       for Hyperbolic Frequency Modulated Waveforms" NRL/FR/7140--94-9713,
%       July, 1994
% Programmer:         J. Paquin Fabre
%               Neptune Sciences, Inc.
% February 1998
%
%Last Modified: 14 February 1998
%Peter Smith

fmin = freqmin;
fmax = freqmax;
fwid = fmax-fmin;
dt=totaltime/fftsize;

binmin = floor(pulsetimemin/dt);
binmax = floor(pulsetimemax/dt);
binlength = binmax - binmin;

n=0:dt:totaltime-dt;

tauc=pulsetimemax - pulsetimemin;
```

```
t = [pulsetimemin:dt:pulsetimemax];

%setting up waveform
A=1.; % arbitrary constant amplitude
phit=((-2*pi*fmin*fmax*tauc)/fwid).*log(fmax*tauc-fwid.*t);
st = A.*exp(i.*phit);

stpadded=[zeros(1,binmin) st zeros(1,fftsize - binmax-1)];

%output
timescale = n;
hfmwave = stpadded;
```

```
%file: inputchange.m
%
%This file allows the user to see and change certain input variables
%from the variables that were chosen from the input file
%
%
%
%
%Last Modified: 14 January 1998
%            Peter Smith
%
%



fprintf('No entry uses the default value shown.');

fcent
changefcent=input('What is the new frequency? ');
if changefcent ~= []
  fcent = changefcent;
end %if fcent

pulsemin
changepulsemin=input('What is the new signal minimum time?(sec) >> ');
if changepulsemin ~= []
  pulsemin = changepulsemin;
end %if pulsemin

pulsemax
changepulsemax=input('What is the new signal maximum time?(sec) >> ');
if changepulsemax ~= []
  pulsemax = changepulsemax;
end %if pulsemax

fftlength
changefftlength=input('What is the new fftlength? >> ');
if changefftlength ~= []
  fftlength = changefftlength;
end %if fftlength

timemax
```

```
changetimemax=input('What is the new maximum time? >> ');
if changetimemax ~= []
  timemax = changetimemax;
end %if timemax

freqmin
changefreqmin=input('What is the new frequency maximum? >>');
if changefreqmin ~= []
  freqmin = changefreqmin;
end %if freqmin

freqmax
changefreqmax=input('What is the new frequency minimum? >>');
if changefreqmax ~= []
  freqmax = changefreqmax
end %if freqmax
```

```
function[timescale,fmwave] = fm(pulsetimemin,pulsetimemax,fftlength,time,...
                                freqmin,freqmax)
%function[timescale,fmwave] = fm(pulsetimemin,pulsetimemax,fftlength,time,...
%                        freqmin,freqmax)
%
%This function will return a time domain fm pulse back to
%the calling program.
%
%
%Input:         pulsetimemin,pulsetimemax,fcent,fftlength,time,freqmin,freqmax,
fftlength,dfreq
%               pulsetimemin - the begining time of the pulse
%               pulsetimemax - the ending time of the pulse (pulselength =max-min)
%               fcent - the frequency of the pulse
%               fftlength - the number of points used in the calling program for
%                     vectors in both time and frequency domain.
%               time - the length of time related to last bin in time domain
%                 timestep = time/fftlength
%               freqmin -the lower frequency of the pulse
%               freqmax -the upper frequency of the pulse
%               fftlength -the number of points used in esl programs.
%               dfreq -the step in frequency for each fft point step
%
%
%Output:        fmwave
%               fmwave- frequency modulated time domain pulse of fcent freqency
%
%Formula:
%From:Fundamentals of Acoustic Field Theory and Space - Time Signal Processing.
%By Lawrence J. Ziomek
%
%Created: 13 January 1998
%Peter E. Smith
%
%Last Modified: 15 February 1998
%Peter E. Smith
%

%calculating variables
pulsetime = pulsetimemax - pulsetimemin;
w0 =  2*pi*freqmin;
alpha = pi*(freqmax-freqmin)/pulsetime;
```

```
time
n=   1:fftlength;
dt = time/fftlength;
N= n*dt;
%caculating pulse
binmin = floor(pulsetimemin/dt);
binmax = floor(pulsetimemax/dt);
binlength = binmax - binmin;
t=[0:dt:dt*binlength];

lfm = real(exp(j*(w0.*t + alpha*(t.^2))));
wave = [zeros(1,binmin) lfm zeros(1,fftlength - binmax -1)];

%output
fmwave = wave;
timescale = N;

%troubleshooting
%fmf = fft(fmwave);

%figure(1)
%plot(N,fmwave)

%figure(2)
%maxfmf = max(abs(fmf));
%plot(abs(fmf))
%axis([3300 3700 0 maxfmf])
%box = [zeros(1,3399) ones(,201) zeros(1,16384-3600)];

%fmmod = fmf.*box;
%mod = real(ifft(fmfmod));

%figure(3)
%plot(abs(fmfmod));
%maxfmfmod = max(abs(fmfmod));
%axis([3300 3700 0 maxfmfmod])

%figure(4)
%plot(N,mod);

%maxnorm = max(fmwave);
%maxmod = max(mod);
%10*log10(mod/fmwave)
```

```
function[mismatchloss,  TL]= mml(rawsig, output, fftsize)
%function[mismatchloss,  TL]= mml(rawsig, output, fftsize)
%
%This function calculates the mismatch loss from the inputs.
%
%Input:          rawsig, otr (freq domain zero padded)
%                rawsig -the sonar pulse being evaluated
%                otr -     the ocean transfer function
%                fftsize - the size of fft being used
%
%Output:         mismatchloss, TL
%                mismatchloss- the peak processing signal loss from correlation due
%                  to energy spreading
%                TL - the comparison
%
%Created: February 14, 1998
%        Peter Smith
%
%Last Modified: February 14, 1998
%        Peter Smith

rawtime = real(ifft(rawsig,fftsize));
timeseries = real(ifft(output,fftsize));

%energy of the original outgoing waveform
Eout = sum(rawtime.^2);

%energy of the stretched pulse
Enospread = sum(timeseries.^2);

r = (Enospread/Eout);

numberotrs = max(size(r));
timereduced = [];

for o=1:numberotrs
  eachreduced = (r(o).^(1/2))*rawtime;
  timereduced = [timereduced eachreduced];
end %for o

%MML
actualcorr = correlate(rawtime,timeseries );
idealcorr = correlate(rawtime,timereduced );
```

```
%plotting correlation output
%figure
%plot(actualcorr(:,1).^2)
%figure
%plot(idealcorr(:,1).^2)


peakspread = max(actualcorr.^2);
peaknospread = max(idealcorr.^2);

absorbTL=-10*log10(r);
mml =-10*log10(peakspread./peaknospread);


%output
mismatchloss = mml;
TL = absorbTL;
```

```
function[mml2way,tl2] = plotmml(rawfreqs,otr,otr2,freqmin,freqmax,fftsize,...
                        dfreq,ranges,scatter)
%function[mml2way,tl2] = plotmml(rawfreqs,otr,otr2,freqmin,freqmax,fftsize,...
%                         dfreq,ranges,scatter)
%
%This program will calculate the 2 way mml and tl for the the two OTR's chosen.
%
%Input:         rawfreqs,otr,freqmin,freqmax,fftsize,dfreq,ranges,pulsetitle,fileid
%               rawfreqs - the initial signal pulse.
%               otr - the ocean transfer function.
%               otr2 - the second ocean transfer function.
%               freqmin - the minimum frequency of otr.
%               freqmax - the maximum frequency of otr.
%               fftsize - the size of fft being used in program.
%               dfreq - the frequency step size in fft.
%               ranges - the ranges relating to the coulumns of the otr for plotting.
%               pulsetitle - the name of the initial signal pulse for title purposes.
%               fileid - the name of the file being used for title purposes.
%               scatter - the scattering function
%
%Output:        2 plots. ESL , TL.
%
%Uses:          outputsig.m, mml.m, avemove.m
%
%Created: 21 January 1998
%Peter Smith
%Last Modified:30 January 1998
%Peter Smith
%

a=min(size(otr));
b=min(size(otr2));

output2way =  outputsig2(rawfreqs,otr(:,a),otr2(:,b),scatter,...
                freqmin,freqmax,fftsize,dfreq);
[mml2, transmissionloss2] = mml(rawfreqs, output2way, fftsize);

%output
mml2way = mml2;
tl2 = transmissionloss2;
```

```
function[ncorr]= normcorr(rawsig, output)
%function[ncorr]= normcorr(rawsig, output)
%
%This function calculates the normalized cross correlation from the inputs.
%
%Input:        rawsig, otr (freq domain zero padded)
%              rawsig -the sonar pulse being evaluated
%              otr -    the ocean transfer function
%              fftsize - the size of fft being used
%
%Output:       ncorr
%              ncorr - the normalized cross correlation
%
%Created: March 4, 1998
%       Peter Smith
%
%Last Modified: March 4, 1998
%       Peter Smith

ideal = real(ifft(rawsig));
timeout = real(ifft(output));

n= size(output);
tempcorr = [];


for o = 1: n(2)
  o
  a = sum(xcorr(ideal, timeout(:,o)));
  u = sum(ideal.^2);
  v = sum(timeout(:,o).^2);
  corrco = (a/sqrt(u+v));
  tempcorr = [tempcorr a];
end %for o

%output
ncorr = tempcorr;
```

```
function[otr] = otrmatload;
%function{otr] = otrmatload;
%
%This function returns the otr function that has been previously saved
%in matlab MAT file format.
%
%Remember to note if a complete otr was not saved for a particular run.
%
%An incomplete OTR may cause inacurate results or kill the main program
%if the changes in the the basic input information has not been modified.
%
%Note: The MAT file saves the variable name of the matrix along with the
%     Matrix.  This program assumes that the matrix name saved is otr.
%
%Input:         none
%
%Output:        otr
%               otr - The ocean transfer function for the ranges chosen
%
%Created: 13 January 1998
%Peter Smith
%
%Last Modified: 15 January 1998
%Peter Smith


%getting save function
MATfilename = input('What is the file name of the MAT file? (use ') >> ');
load(MATfilename);
```

```
function[] = otrmatsave(otr);
%function{otr] = otrmatload;
%
%This function saves the otr function.
%
%Remember to note if a complete otr was saved.
%
%An incomplete OTR may cause inacurate results or kill the main program
%if the changes in the the basic input information has not been modified.
%
%Last Modified: January 13 1998
%


MATfilename = input('Save the MAT file as: (use '') ');

save(MATfilename,'otr');
```

```
function [] = otrplot(otr, ranges, freqmin, freqmax, fftlength, dfreq,...
                rangerequest,fileid)
%function [] = otrplot(otr, ranges, freqmin, freqmax, fftlength, dfreq,...
%                 rangerequest,fileid)
%
%This function will plot the ocean transfer function for a particular range
%requested.
%
%inputs:        otr, ranges, freqmin, freqmax, fftlength, dfreq,rangerequest,fileid
%               otr - matrix of ocean transfer functions
%               ranges - the ranges that correlate to the otr matrix
%               freqmin - the lower frequency of the pulse
%               freqmax - the upper frequency of the pulse
%               fftlength - the number of points used in esl programs.
%               dfreq    - the step in frequency for each fft point step
%               rangerequest - the ranges selected for plotting
%               fileid - the title of the file name used for plotting
%
%
%outputs:       no numerical outputs
%               The output is a plot of the OTR function vrs frequency.
%
%Created: 13 January 1998
%Peter Smith
%
%Last Modified: 30 January 1998
%Peter Smith
%


%Finding the Scale factors for the plots
x = freqmin:dfreq:freqmax;

%Finding the requested OTR's
otrnumber = [];

numrequest = max(size(rangerequest));
 for m = 1:numrequest;
   otrfound = find(rangerequest(m) == ranges);
   if isempty(otrfound)
     warning = ['The following range request index was not found in the ranges:']
     m
    end % if
```

```
    otrnumber = [otrnumber otrfound];
    clear otrfound
  end %for m



%plotting
typeplot = menu('What type of plot do you want?','1 Plot- 1 Graph','Subplots');


if typeplot == 1
  for n = 1:numrequest;
    figure
    clf
    plot(x,abs(otr(:,(otrnumber(:,n)))))

    title([fileid ,' ', num2str(rangerequest(n)/1000),'k OTR']);
    xlabel('Frequency (Hz)')
  end %for n
end % if type 1

if typeplot == 2
  o = numrequest;
  figure
  for p = 1: numrequest;
    subplot(o,1,p)
    plot(x,abs(otr(:,(otrnumber(:,p)))))
    title([fileid ,' ', num2str(rangerequest(p)/1000),'k OTR']);
  end %for p
    xlabel('Frequency (Hz)')
end  if type 2
```

```
function[] = outputplott(rawsig,otr,ranges,freqmin,freqmax, fftlength, dfreq,...
                dtime,rangerequest,fileid)
%function[] = outputplott(rawsig,otr ranges,freqmin,freqmax, fftlength, dfreq,...
%                 dtime,rangerequest,fileid)
%
%This program will movie the output of the ocean transfer function and the input
%pulse. An option is given at the end to run the movie with a chosen number of
%runs.
%
%inputs:        rawsig, otr, ranges, freqmin, freqmax, fftlength, dfreq,dtime
%               rangerequest,fileid
%               rawsig - the input pulse in the frequency domain
%               otr - matrix of ocean transfer functions
%               ranges - the ranges that correlate to the otr matrix
%               freqmin - the lower frequency of the pulse
%               freqmax - the upper frequency of the pulse
%               fftlength - the number of points used in esl programs.
%               dfreq - the step in frequency for each fft point step
%               dtime - the time step for each fft bin
%               rangerequest - the ranges selected for plotting
%               fileid - the title of the file name used for plotting
%
%outputs        :no numerical outputs
%               The output is a movie of the time domain of the output.
%
%Uses: outputsig.m
%
%
%Created: 30 January 1998
%Peter Smith
%
%Last Modified: 26 February 1998
%Peter Smith
%


%Finding the Scale factors for the plots
maxtime = dtime * (fftlength-1);
x = 0:dtime:maxtime;

%Finding the requested OTR's
otrnumber = [];
```

```
numrequest = max(size(rangerequest));
 for m = 1:numrequest;
  otrfound = find(rangerequest(m) == ranges);
  if isempty(otrfound)
   warning = ['The following range request index was not found in the ranges:']
   m
  end % if
  otrnumber = [otrnumber otrfound];
  clear otrfound
 end %for m

outputfreq = outputsig(rawsig,otr,freqmin,freqmax,fftlength,dfreq);
outputtime = flipud(real(ifft(outputfreq)));

%plotting

numberframes = numrequest;
M = moviein(numberframes);%movie stored in (large) matrix M


for n = 1:numberframes;
 plot(x/1000, outputtime(:,(otrnumber(:,n))))

 setting axis values for normalizing movie
 if n == 1
  maxaxis = max(outputtime(:,(otrnumber(:,n))));
 end %if n
 axis([0 maxtime/1000 0 maxaxis]);
 title([fileid ,' ', num2str(rangerequest(n)/1000),'k Output']);
 xlabel('Time')
 M(:,n) = getframe;
end %for n
runmovie = 1;

while runmovie == 1
 title([fileid ,' ', num2str(rangerequest(1)/1000),'k to ',...
       num2str(rangerequest(numberframes)/1000),'k Output']);
 runthrough = input('How many run throughs? >> ');
 fps = input('Number of frames per second: >> ');
 movie(M,runthrough,fps)
 runmovie = menu('Run Movie?','Yes','No');
 clf
end %while runmovie
```

```
function[] = outputplotf(rawsig, otr, ranges, freqmin, freqmax, fftlength,...
                  dfreq, rangerequest, fileid)
%function[] = outputplotf(rawsig, otr, ranges, freqmin, freqmax, fftlength,...\
%                  dfreq, rangerequest, fileid)
%
%This program will plot the output of the ocean transfer function and the input
%pulse
%
%inputs:        rawsig, otr, ranges, freqmin, freqmax, fftlength, dfreq, rangerequest,
%               fileid.
%               rawsig - the input pulse in the frequency domain
%               otr - matrix of ocean transfer functions
%               ranges - the ranges that correlate to the otr matrix
%               freqmin - the lower frequency of the pulse
%               freqmax - the upper frequency of the pulse
%               fftlength - the number of points used in esl programs
%               dfreq    - the step in frequency for each fft point step
%               rangerequest - the ranges selected for plotting
%               fileid - the title of the file name used for plotting
%
%outputs:       no numerical outputs
%               The output is a plot of the output in the frequency domain.
%
%Uses: outputsig.m
%
%
%Created: 30 January 1998
%Peter Smith
%
%Last Modified: 1 February 1998
%Peter Smith
%


%Finding the Scale factors for the plots
x = 0:dfreq:(fftlength-1)*dfreq;

%Finding the requested OTR's
otrnumber = [];

numrequest = max(size(rangerequest));
 for m = 1:numrequest;
   otrfound = find(rangerequest(m) == ranges);
```

```
    if isempty(otrfound)
      warning = ['The following range request index was not found in the ranges:']
      m
     end % if
    otrnumber = [otrnumber otrfound];
    clear otrfound
  end %for m

outputfreq = outputsig(rawsig,otr,freqmin,freqmax,fftlength,dfreq);


%plotting
typeplot = menu('What type of plot do you want?','1 Plot- 1 Graph','Subplots');

if typeplot == 1
  for n = 1:numrequest;
    figure
    clf
    plot(x,abs(outputfreq(:,(otrnumber(:,n)))))

    axis([freqmin freqmax 0 max(abs(outputfreq(:,(otrnumber(:,n)))))]);
    title([fileid ,' ', num2str(rangerequest(n)/1000),'k Output']);
    xlabel('Frequency Hz)')
  end %for n
end % if type 1

if typeplot == 2
  o = numrequest;
  figure
  for p = 1: numrequest;
    subplot(o,1,p)
    plot(x,abs(outputfreq(:,(otrnumber(:,p)))))

    axis([freqmin freqmax 0 max(abs(outputfreq(:,(otrnumber(:,p)))))]);
    title([fileid ,' ', num2str(rangerequest(p)/1000),'k Output']);
  end %for p
    xlabel('Frequency (Hz)')
end % if type 2
```

```
function[] = outputplott(rawsig,otr,ranges,freqmin,freqmax, fftlength, dfreq,...
                dtime,rangerequest,fileid)
%function[] = outputplott(rawsig,otr ranges,freqmin,freqmax, fftlength, dfreq,...
%                  dtime,rangerequest,fileid)
%
%This program will plot the output of the ocean transfer function and the input
%pulse
%
%inputs:        rawsig, otr, ranges, freqmin, freqmax, fftlength, dfreq,dtime
%               rangerequest,fileid
%               rawsig - the input pulse in the frequency domain
%               otr - matrix of ocean transfer functions
%               ranges - the ranges that correlate to the otr matrix
%               freqmin -  the lower frequency of the pulse
%               freqmax - the upper frequency of the pulse
%               fftlength - the number of points used in esl programs.
%               dfreq - the step in frequency for each fft point step
%               dtime - the time step for each fft bin
%               rangerequest - the ranges selected for plotting
%               fileid - the title of the file name used for plotting
%
%outputs:       no numerical outputs
%               The output is a plot of the output in the time domain.
%
%Uses: outputsig.m
%
%
%Created: 30 January 1998
%Peter Smith
%
%Last Modified: 25 February 1998
%Peter Smith
%


%Finding the Scale factors for the plots
maxtime = dtime * (fftlength-1);
x = 0:dtime:maxtime;

%Finding the requested OTR's
otrnumber = [];

numrequest = max(size(rangerequest));
```

```
  for m = 1:numrequest;
    otrfound = find(rangerequest(m) == ranges);
    if isempty(otrfound)
      warning = ['The following range request index was not found in the ranges:']
      m
     end % if
    otrnumber = [otrnumber otrfound];
    clear otrfound
  end %for m

outputfreq = outputsig(rawsig,otr,freqmin,freqmax,fftlength,dfreq);
outputtime = flipud(real(ifft(outputfreq)));

%plotting
typeplot = menu('What type of plot do you want?','1 Plot- 1 Graph','Subplots');
shiftsize = input('What shift do you want? >> ');

if shiftsize > 0
  temp1 = outputtime(1:shiftsize,:);
  temp2 = outputtime(shiftsize+1:fftlength,:);
  outputtime =[temp2;temp1];
end % if shift

if typeplot == 1
  for n = 1:numrequest;
    figure
    clf
    plot(x/1000, outputtime(:,(otrnumber(:,n))))

    axis([0 maxtime/1000 0 max(outputtime(:,(otrnumber(:,n)))) ]);
    title([fileid ,' ', num2str(rangerequest(n)/1000),'k Output']);
    xlabel('Time')
  end %for n
end % if type 1

if typeplot == 2
  o = numrequest;
  figure
  for p = 1: numrequest;
    subplot(o,1,p)
    plot(x/1000, outputtime(:,(otrnumber(:,p))))

    axis([0 maxtime/1000 0 max(outputtime(:,(otrnumber(:,1)))) ]);
```

```
   axis off
  end %for p
   xlabel('Time')
end % if type 2

function[outputsignal] = outputsig(rawsig,otr,freqmin,freqmax,fftsize,dfreq);
%function[outputsignal] = outputsig(rawsig,otr,freqmin,freqmax,fftsize,dfreq);
%
%This program will create the output signal in the frequency domain.
%
%input:          rawsig,otr,freqmin,freqmax,fftsize,dfreq
%                rawsig - the sonar pulse in zeropadded freq domain
%                otr -      the unpadded ocean transfer function
%                freqmin - the minimum frequency in the otr
%                freqmax - the maximum frequency in the otr
%                fftsize - the size of fft being used
%                dfreq -   the size of the frequency interval
%
%Output:         outputsignal
%                outputsignal - the output signal in the frequency domain;
%
%Uses :          zeropadding.m
%
%Created: 14 January 1998
%        Peter Smith
%
%Last Modified: 14 February 1998
%             Peter Smith
%
%

%zeropadding the otr function to place frequencies in correct bins
paddedotr = zeropadding(otr,freqmin,freqmax,fftsize,dfreq);

%troubleshooting
%size(paddedotr)
%size(rawsig)

%finding number of otrs in matrix
%(works unless your number of otrs > fft size)
numberotrs = min(size(otr));

outputmatrix = [];
```

```
for n = 1:numberotrs
  output = paddedotr(:,n) .* rawsig;
  outputmatrix = [outputmatrix output];
end % for n
%output
outputsignal = outputmatrix;
function[output2signal] = outputsig2(rawsig,otr,otr2,scatter,...
                              freqmin,freqmax,fftsize,dfreq);
%function[output2signal] = outputsig2(rawsig,otr,otr2,scatter,...
%                               freqmin,freqmax,fftsize,dfreq);
%
%This program will create the 2 way output signal in the frequency domain.
%
%input:rawsig,otr,otr2,scatter,freqmin,freqmax,fftsize,dfreq
%       rawsig - the sonar pulse in zeropadded freq domain
%       otr -     the unpadded ocean transfer function (not matrix)
%       otr2-    the second unpadded ocean transfer function (not matrix)
%       scatter- the scattering function in zeropadded freq domain
%       freqmin - the minimum frequency in the otr
%       freqmax - the maximum frequency in the otr
%       fftsize - the size of fft being used
%       dfreq -   the size of the frequency interval
%
%Output: output2signal
%       output2signal - the output signal in the frequency domain;
%
%Uses :  zeropadding.m
%
%Created: 14 January 1998
%       Peter Smith
%
%Last Modified: 14 February 1998
%            Peter Smith
%
%


%zeropadding the otr function to place frequencies in correct bins
paddedotr = zeropadding(otr,freqmin,freqmax,fftsize,dfreq);
paddedotr2 = zeropadding(otr2,freqmin,freqmax,fftsize,dfreq);


%troubleshooting
```

```
%size(paddedotr)
%size(rawsig)

%finding number of otrs in matrix
%(works unless your number of otrs > fft size)

output = paddedotr .* paddedotr2 .* scatter .* rawsig;

%output
output2signal = output;
```

```
function[] = plotcompesl(rawfreqs,otr,otr2,freqmin,freqmax,fftsize,dfreq,ranges,...
                    pulsetitle,fileid, tmin,tmax)
%function[] = plotcompesl(rawfreqs,otr,otr2,freqmin,freqmax,fftsize,dfreq,ranges,...
%                  pulsetitle,fileid, tmin, tmax)
%
%This program will calculate and plot the ESL for the OTR matrix chosen.
%
%Input:         rawfreqs,rawtime,otr,otr2,freqmin,freqmax,fftsize,dfreq,
%               ranges,pulsetitle,fileid
%               rawfreqs - the initial signal pulse in frequency.
%               otr - the ocean transfer function.
%               otr2 - the comparison ocean transfer function.
%               freqmin - the minimum frequency of otr.
%               freqmax - the maximum frequency of otr.
%               fftsize - the size of fft being used in program.
%               dfreq - the frequency step size in fft.
%               ranges - the ranges relating to the coulumns of the otr for plotting.
%               pulsetitle - the name of the initial signal pulse for title purposes.
%               fileid - the name of the file being used for title purposes.
%               tmin - the minimum start time for fft from input file
%               tmax - the maximum start time for fft from input file
%
%Output:        2 plots. ESL , TL.
%
%Uses: outputsig.m, esl.m, avemove.m
%
%Created: 14 February 1998
%Peter Smith
%Last Modified: 11 March 1998
%Peter Smith
%

comparepulse = menu('Do You Want to Compare Pulse Types','Yes','No');

if comparepulse == 1
   output =  outputsig(rawfreqs,otr,freqmin,freqmax,fftsize,dfreq);
   [eslloss, transmissionloss] = esl(rawfreqs,output,fftsize);
   [trunkranges,eslave,derranges,der1st] = avemove(ranges,eslloss);

    pulseshape;

   output2 =  outputsig(rawfreqs,otr2,freqmin,freqmax,fftsize,dfreq);
   [esl2loss, transmission2loss] = esl(rawfreqs,output2,fftsize);
```

```
    [trunk2ranges,esl2ave,der2ranges,der1st2] = avemove(ranges,esl2loss);
end %if

if comparepulse == 2
  output =  outputsig(rawfreqs,otr,freqmin,freqmax,fftsize,dfreq);
    [eslloss, transmissionloss] = esl(rawfreqs,output,fftsize);
    [trunkranges,eslave,derranges,der1st] = avemove(ranges,eslloss);
  output2 =  outputsig(rawfreqs,otr2,freqmin,reqmax,fftsize,dfreq);
    [esl2loss, transmission2loss] = esl(rawfreqs,output2,fftsize);
    [trunk2ranges,esl2ave,der2ranges,der1st2] = avemove(ranges,esl2loss);
end %if

 esl3loss = eslloss - esl2loss;
 esl3ave = eslave - esl2ave;


    figure
    subplot(3,1,1)
    plot(ranges/1000,eslloss)
    axismax = max(ranges)/1000;
    hold on
    plot(trunkranges/1000,eslave,'x')
    axis([0 axismax 0 10]);
    title('(a)')

    M = gca;
    set(M,'YTick',[0:2:10]);
    grid on

    subplot(3,1,2)
    plot(ranges/1000,esl2loss)
    axismax = max(ranges)/1000;
    hold on
    plot(trunkranges/1000,esl2ave,'x')
    axis([0 axismax 0 10]);
    title('(b)')
    ylabel('dB')

    N = gca;
    set(N,'YTick',[0:2:10]);
    grid on

    subplot(3,1,3)
```

```
plot(ranges/1000,esl3loss)
axismax = max(ranges)/1000;
hold on
plot(trunkranges/1000,esl3ave,'x')
title('(c)')

maxloss = max(esl3loss);
minloss = min(esl3loss);

if (maxloss < 5) & (minloss > -5)
  axis([0 axismax -5  5]);
  O = gca;
  set(O,'YTick',[-4:2:4]);
 else
  axis([0 axismax minloss minloss+10]);
  a = floor(minloss);
  O = gca;
  b = rem(a,2);
  if b == 0
    c = a;
    d = a+10;
   else
    c = a+1;
    d = a+11;
  end % if b
  set(O,'YTick',[c:2:d]);
end %if max/min
grid on
xlabel('Range (KM)')
```

```
function[] = plotcompmml(rawfreqs,otr,otr2,freqmin,freqmax,fftsize,dfreq,ranges,...
                    pulsetitle,fileid,tmin,tmax)
%function[] = plotcompmml(rawfreqs,otr,otr2,freqmin,freqmax,fftsize,dfreq,ranges,...
%                  pulsetitle,fileid,tmin,tmax)
%
%This program will calculate and plot the MML for the OTR matrix chosen.
%
%Input:          rawfreqs,rawtime,otr,otr2,freqmin,freqmax,fftsize,dfreq,
%                ranges,pulsetitle,fileid
%                rawfreqs - the initial signal pulse in frequency.
%                otr - the ocean transfer function.
%                otr2 - the comparison ocean transfer function.
%                freqmin - the minimum frequency of otr.
%                freqmax - the maximum frequency of otr.
%                fftsize - the size of fft being used in program.
%                dfreq - the frequency step size in fft.
%                ranges - the ranges relating to the coulumns of the otr for plotting.
%                pulsetitle - the name of the initial signal pulse for title purposes.
%                fileid - the name of the file being used for title purposes.
%                tmin - the minimum start time for fft from input file
%                tmax - the maximum start time for fft from input file
%
%Output:        2 plots. ESL , TL.
%
%Uses: outputsig.m, mml.m, avemove.m
%
%Created: 14 February 1998
%Peter Smith
%Last Modified: 12 March 1998
%Peter Smith
%

centerfreq = (freqmin + freqmax)/2;


comparepulse = menu('Do You Want to Compare Pulse Types','Yes','No');

if comparepulse == 1
  output =  outputsig(rawfreqs,otr,freqmin,freqmax,fftsize,dfreq);
    [mmlloss, transmissionloss] = mml(rawfreqs,output,fftsize);
    [trunkranges,mmlave,derranges,der1st] = avemove(ranges,mmlloss);

  pulseshape;
```

```
  output2 =  outputsig(rawfreqs,otr2,freqmin,freqmax,fftsize,dfreq);
    [mml2loss, transmission2loss] = mml(rawfreqs,output2,fftsize);
    [trunk2ranges,mml2ave,der2ranges,der1st2] = avemove(ranges,mml2loss);
end %if

if comparepulse == 2
  output =  outputsig(rawfreqs,otr,freqmin,freqmax,fftsize,dfreq);
    [mmlloss, transmissionloss] = mml(rawfreqs,output,fftsize);
    [trunkranges,mmlave,derranges,der1st] = avemove(ranges,mmlloss);
  output2 =  outputsi(rawfreqs,otr2,freqmin,freqmax,fftsize,dfreq);
    [mml2loss, transmission2loss] = mml(rawfreqs,output2,fftsize);
    [trunk2ranges,mml2ave,der2ranges,der1st2] = avemove(ranges,mml2loss);
end %if

 mml3loss = mmlloss - mml2loss;
 mml3ave = mmlave - mml2ave;


    figure
    subplot(3,1,1)
    plot(ranges/1000,mmlloss)
    axismax = max(ranges)/1000;
    hold on
    plot(trunkranges/1000,mmlave,'x')
    %ylabel('dB (x-moving 5 point ave)')
    %xlabel('Range (KM)')
    title('(a)')
    axis([0 axismax 0 10]);

    M = gca;
    set(M,'YTick',[0:2:10]);
    grid on

    subplot(3,1,2)
    plot(ranges/1000,mml2loss)
    axismax = max(ranges)/1000;
    hold on
    plot(trunkranges/1000,mml2ave,'x')
    %ylabel('dB (x-moving 5 point ave)')
    %xlabel('Range (KM)')
    title('(b)')
    axis([0 axismax 0 10]);
```

```
N = gca;
set(N,'YTick',[0:2:10]);
grid on

subplot(3,1,3)
plot(ranges/1000,mml3loss)
axismax = max(ranges)/1000;
hold on
plot(trunkranges/1000,mml3ave,'x')
%ylabel('dB (x-moving 5 point ave)')
%xlabel('Range (KM)')
title('(c)')

maxloss = max(mml3loss);
minloss = min(mml3loss);

if (maxloss < 5) & (minloss > -5)
  axis([0 axismax -5  5]);
  O = gca;
  set(O,'YTick',[-4:2:4]);
 else
  axis([0 axismax minloss minloss+10]);
  a = floor(minloss);
  O = gca;
  b = rem(a,2);
  if b == 0
    c = a;
    d = a+10;
   else
    c = a+1;
    d = a+11;
  end % if b
  set(O,'YTick',[c:2:d]);
end %if max/min
grid on
```

```
function[] = plotesl(rawfreqs,otr,freqmin,freqmax,fftsize,dfreq,ranges,...
                pulsetitle,fileid)
%function[] = plotesl(rawfreqs,otr,freqmin,freqmax,fftsize,dfreq,ranges,...
%                 pulsetitle,fileid)
%
%This program will calculate and plot the esl and tl for the OTR matrix chosen.
%
%Input:          rawfreqs,otr,freqmin,freqmax,fftsize,dfreq,ranges,pulsetitle,fileid
%                rawfreqs - the initial signal pulse.
%                otr - the ocean transfer function.
%                freqmin - the minimum frequency of otr.
%                freqmax - the maximum frequency of otr.
%                fftsize - the size of fft being used in program.
%                dfreq - the frequency step size in fft.
%                ranges - the ranges relating to the coulumns of the otr for plotting.
%                pulsetitle - the name of the initial signal pulse for title purposes.
%                fileid - the name of the file being used for title purposes.
%
%Output:         2 plots. ESL , TL.
%
%Uses: outputsig.m, esl.m, avemove.m
%
%Created: 21 January 1998
%Peter Smith
%Last Modified:12 March 1998
%Peter Smith
%

plottype = menu('Do you want subplots?','Yes','No');




 output =  outputsig(rawfreqs,otr,freqmin,freqmax,fftsize,dfreq);
   [eslloss, transmissionloss] = esl(rawfreqs,output,fftsize);
   [trunkranges,eslave,derranges,der1st] = avemove(ranges,eslloss);

if plottype == 1
   figure(1)
   plotloc = input('What plot location do you want? >> ');

   subplot(2,2,plotloc)
```

```
    plot(ranges/1000,eslloss)
    axismax = max(ranges)/1000;
    hold on
    plot(trunkranges/1000,eslave,'x')

    %ylabel('dB')
    ylabel('ESL (dB)')
    xlabel('Range (KM)')
    axis([0 axismax 0 10]);

  else

    I= find(der1st < 0);
    zero1 = ranges(I(1)+1)/1000;

    figure
    plot(ranges/1000,eslloss)
    axismax = max(ranges)/1000;
    hold on
    plot(trunkranges/1000,eslave,'x')
    title([fileid, ' ESL - ',pulsetitle])
    %ylabel('dB (x-moving 5 point ave)')
    ylabel('ESL (dB)')
    xlabel('Range (KM)')
    axis([0 axismax 0 10]);

    figure
    plot(ranges/1000,-transmissionloss)
    title([fileid, ' TL - ',pulsetitle])    ylabel('TL dB')
    xlabel('Range (KM)')

    figure
    plot(derranges/1000,der1st)
    title([fileid,' - ',pulsetitle ' 1st Derivative of 5 Point Average'])
    ylabel('Slope of Ave. - DB')
    xlabel(['Range (KM) 1st Zero = ',num2str(zero1),' KM'])
end %if plottype

%output of values
%eslloss'
%eslave'
```

```
function[] = plotmml(rawfreqs,otr,freqmin,freqmax,fftsize,dfreq,ranges,...
                     pulsetitle,fileid)
%function[] = plotmml(rawfreqs,otr,freqmin,freqmax,fftsize,dfreq,ranges,...
%                  pulsetitle,fileid)
%
%This program will calculate and plot the MML for the OTR matrix chosen.
%
%Input:         rawfreqs,rawtime,otr,freqmin,freqmax,fftsize,dfreq,
%               ranges,pulsetitle,fileid
%               rawfreqs - the initial signal pulse in frequency.
%               otr - the ocean transfer function.
%               freqmin - the minimum frequency of otr.
%               freqmax - the maximum frequency of otr.
%               fftsize - the size of fft being used in program.
%               dfreq - the frequency step size in fft.
%               ranges - the ranges relating to the coulumns of the otr for plotting.
%               pulsetitle - the name of the initial signal pulse for title purposes.
%               fileid - the name of the file being used for title purposes.
%
%Output:        2 plots. ESL , TL.
%
%Uses: outputsig.m, esl.m, avemove.m
%
%Created: 14 February 1998
%Peter Smith
%Last Modified: 14 February 1998
%Peter Smith
%

output =  outputsig(rawfreqs,otr,freqmin,freqmax,fftsize,dfreq);
[mmlloss, transmissionloss] = mml(rawfreqs,output,fftsize);
[trunkranges,mmlave,derranges,der1st] = avemove(ranges,mmlloss);


  figure
  plot(ranges/1000,mmlloss)
  axismax = max(ranges)/1000;
  hold on
  plot(trunkranges/1000,mmlave,'x')
  title([fileid, ' MML - ',pulsetitle])
  ylabel('dB (x-moving 5 point ave)')
  xlabel('Range (KM)')
  axis([0 axismax 0 10]);%
```

```
figure
plot(ranges/1000,-transmissionloss)
title([fileid, ' TL - ',pulsetitle])
ylabel('TL dB')
xlabel('Range (KM)')

figure
plot(derranges/1000,der1st)
title([fileid,' - ',pulsetitle ' 1st Derivative of 5 Point Average'])
ylabel('Slope of Ave. - DB')
xlabel('Range (KM)')
```

```
function[] = plotpulsef(rawsig,freqmin,freqmax,dfreq,fftsize,pulsetitle)
%function[] = plotpulsef(rawsig,freqmin,freqmax,dfreq,fftsize,pulsetitle)
%
%This program will plot the pulse in the frequency domain for the
%frequencies of interest.
%
%Input:      rawsig,freqmin,freqmax,dfreq,fftsize
%            rawsig - the frequency domain of the sonar pulse of interest
%            freqmin - the minimum frequency of interest
%            freqmax - the maximum frequency of interest
%            dfreq - the frequency step of the sampling
%            fftsize - the size of the fft's being used
%            pulsetitle - the name of the pulse type for labeling
%
%Output:     The plot of the amplitude of the pulse versus the frequencies
%            of interst.
%
%
%Created: 14 January 1998
%            Peter Smith
%Last Modified: 19 January 1998
%            Peter Smith
%

maxf = dfreq*(fftsize-1);

n = 0:dfreq:maxf;


plotsig = abs(rawsig);

ampmax = max(plotsig);

figure
plot(n,plotsig)
axis([freqmin freqmax 0 ampmax]);
title(pulsetitle)
xlabel('Frequency')
```

```
function[] = plotpulsef(rawsig,dtime,fftsize,pulsetitle)
%function[] = plotpulsef(rawsig,freqmin,freqmax,dfreq,fftsize)
%
%This program will plot the pulse in the frequency domain for the
%frequencies of interest.
%
%Input:          rawsig, dtime, fftsize, pulsetitle
%                rawsig - the frequency domain of the sonar pulse of interest
%                dtime - the time step for each fft bin
%                fftsize - the size of fft being used in program
%                pulsetitle - the name of the pulse for labeling purposes
%
%Output:         The plot of the amplitude of the pulse versus the frequencies
%                of interst.
%
%
%Created: 14 January 1998
%            Peter Smith
%Last Modified: 11 March 1998
%                Peter Smith


maxtime = dtime * (fftsize-1);

n = 0:dtime:maxtime;

plotpulsetime = real(ifft(rawsig,fftsize));

shiftsize = input('What shift do you want? >> ');
if shiftsize > 0
  temp1 = plotpulsetime(1:shiftsize,:);
  temp2 = plotpulsetime(shiftsize+1:fftsize,:);
  plotpulsetime = [temp2;temp1];
end % if shift

figure
plot(n,plotpulsetime);
title(pulsetitle)
xlabel('Time')
```

```
%file :  pulseshape.m
%
%This program returns frequency domain information of the pulse type chosen
%Each pulse type is defined in a seperate function.
%The information for the frequencies is from the input file that should
%have been read by chosing the Run input function.
%The pulse type functions could call functions from the
%Signal Processing Toolbox.
%
%Created: 12 January 1998
%Peter Smith
%
%Last Modified:12 March 1998
%Peter Smith
continue = 0;

if continue == 0
   pulsetype = menu('What type of pulse do you want?','Bartlett','Blackman',...
               'Box','Hamming','Hanning','Triangle','CW','LFM','HFM');

 if (pulsetype == 1)
   rawfreqs = bart(freqmin, freqmax, fftsize, dfreq);
   pulsetitle = 'Bartlett';
   rawtime = ifft(rawfreqs);
 end %if 1
 if (pulsetype == 2)
   rawfreqs = black(freqmin, freqmax, fftsize, dfreq);
   pulsetitle = 'Blackman';
   rawtime = ifft(rawfreqs);
 end %if 2
 if (pulsetype == 3)
   rawfreqs = boxpulse(freqmin, freqmax, fftsize, dfreq);
   pulsetitle = 'Box';
   rawtime = ifft(rawfreqs);
 end %if 3
 if (pulsetype == 4)
   rawfreqs = ham(freqmin, freqmax, fftsize, dfreq);
   pulsetitle = 'Hamming';
   rawtime = ifft(rawfreqs);
 end %if 1
 if (pulsetype == 5)
   rawfreqs = han(freqmin, freqmax, fftsize, dfreq);
   pulsetitle = 'Hanning';
```

```
    rawtime = ifft(rawfreqs);
  end %if 2
  if (pulsetype == 6)
    rawfreqs = tri(freqmin, freqmax, fftsize, dfreq);
    pulsetitle = 'Triangle';
    rawtime = ifft(rawfreqs);
  end %if 6
  if (pulsetype == 7)
    pulsetitle = 'CW';
    totaltime = (tmax - tmin)/1000;
    pulsetimemin = input('What is starting time for cw pulse? >> ');
    pulsetimemax = input('What is finishing time for cw pulse? >> ');
    [timescale, cwwave] = cw(pulsetimemin,pulsetimemax,centerfreq,...
                        fftsize,totaltime);
    rawfreqs = fft(cwwave)';
    rawtime = cwwave;
  end %if 7
  if (pulsetype == 8)
    pulsetitle = 'LFM';
    totaltime = (tmax - tmin)/1000;
    pulsetimemin = input('What isstarting time for LFM pulse? >> ');
    pulsetimemax = input('What is finishing time for LFM pulse? >> ');
    [timescale, lfmwave] = lfm(pulsetimemin,pulsetimemax,fftsize,totaltime,...
                          freqmin,freqmax);
    rawfreqs = fft(lfmwave)';
    rawtime = lfmwave;
  end %if 8
if (pulsetype == 9)
    pulsetitle = 'HFM';
    totaltime = (tmax - tmin)/1000;
    pulsetimemin = input('What is starting time for HFM pulse? >> ');
    pulsetimemax = input('What is finishing time for HFM pulse? >> ');
    [timescale, hfmwave] = hfm(pulsetimemin,pulsetimemax,...
                        fftsize,totaltime,freqmin,freqmax);
    rawfreqs = fft(hfmwave)';
    rawtime = real(ifft(hfmwave));
  end %if 8
end %if continue
continue = 0;
```

```
%file:readinput.m
%
%This m file will read the input file that is requested from the user
%header information.
%
%Created: 12 January 1998
%Peter Smith
%
%Last Modified: 21 January 1998
%Peter Smith
%
%

%file name input
fileid = input('What run do you want to process? (use '') >> ');

filename = fileid;
inputname = [filename,'.in'];
load(inputname);

data = eval(filename)';

tmin = data(1);
tmax = data(2);
dtime = data(3);
fftsize = data(4);
centerfreq = data(5);
sourcedepth = data(6);
recieverdepth = data(7);
freqmin = data(8);
freqmax = data(9);
rangemax = data(10);
horizontalstep = data(11);
rangeskip = data(12);
rangestart = data(13);
rangefinish = data(14);
depthmax = data(15);
verticlestep = data(16);
depthskip = data(17);
depthstart = data(18);
depthfinish = data(19);
recordsize = data(20);
initialc = data(21);
```

```
%calculation of variables from input data
rangestep = horizontalstep * rangeskip;
totalrange = rangefinish - rangestart;
ranges =  (rangestart+rangestep):rangestep:rangefinish;

depthstep = verticlestep * depthskip;
totaldepth = depthfinish - depthstart;
depths =  depthstart:depthstep:depthfinish;

dfreq = 1/(tmax*0.001 - tmin *0.001);


%getting rid of excess information
clear inputname filename data
```

```
function[trianglefreq]= tri(freqmin, freqmax, fftlength, dfreq)
%function[trianglefreq]= tri(freqmin, freqmax, fftlength, dfreq)
%
%
%inputs:        freqmin,freqmax, fftlength,dfreq
%               freqmin - the lower frequency of the pulse
%               freqmax -the upper frequency of the pulse
%               fftlength -the number of points used in esl programs.
%               dfreq - the step in frequency for each fft point step
%
%outputs:       trianglefreq
%               The output is the frequency domain traingle pulse.
%
%This function uses the signal processing triangle function.

imin = floor(freqmin/dfreq) + 1;
imax = floor(freqmax/dfreq) + 1;
numberpoints = imax - (imin -1);

pulseshape = triang(numberpoints);

freqs = [zeros(imin-1,1);pulseshape;zeros((fftlength-imax),1)];

%output
trianglefreq = freqs;
```

```
function[otrfreqs] = zeropadding(otr,freqmin,freqmax,fftlength,dfreq)
%function[otrfreqs] = zeropadding(otr,freqmin,freqmax,fftlength,dfreq)
%
%This function zeropads the otr for use in evaluation.
%
%input:          otr,freqmin,freqmax,fftsize,dfreq
%                otr    - the matrix of ocean transfer functions
%                freqmin - the minimum frequency of the otr
%                freqmax - the maximum frequency of the otr
%                fftsize- the size of the fft being used
%                dfreq  - the step size between frequencies
%
%output:         otrfreqs
%                otrfreqs - the zeropadded matrix of ocean transfer functions.
%
%Created: 15 December 1997
%Peter Smith
%
%Last Modified: 21 January 1998
%Peter Smith
%
%



%getting bin placement
imin = floor(freqmin/dfreq) + 1;
imax = floor(freqmax/dfreq) + 1;

%getting number of otr's
otrsize = size(otr);
numberotrs = otrsize(2);

%zero padding to place otr at correct frequencies
paddedfreqs = [zeros(imin-1,numberotrs);otr;...
        zeros((fftlength-imax),numberotrs)];

%output
otrfreqs = paddedfreqs;
```

LIST OF REFERENCES

B.S. Adams, " An analysis of the effects of energy spreading loss on low frequency active sonar operations in shallow water," Masters Thesis, Naval Postgraduate School, Monterey, CA, 1996.

L.P. Atkinson, J. J. Singer, and L. J. Pietrafesa, "Volume of summer subsurface intrusions into Onslow Bay, North Carolina," Deep-Sea Research, 27 A, 421-434 ,1980.

T.G. Bell, "Predicting and dealing with energy spreading loss," in Proceedings of a Seminar on Active Sonar Signal Processing, 12 Dec 1989, Naval Undersea Systems Center, New London, Tracor Doc. T90-01-9532-U, 1990.

R. Bonds, *The US War Machine*, New York: Crown, 1987.

P. J. Bucca, J. K. Fulford, and R.W. Nero, "Pre-Experiment environmental characterization for the joint NAVSEA/littoral warfare advanced development system concept validation (SCV-97) tests in Long Bay, S.C." NRL/MR/7182-97-8057, Naval Research Laboratory, Stennis Space Center, 1997.

F. Chan, "Final report on the time spreading and energy spreading losses for the AN/SQS-53C shallow water (9/91) sea test", NUWC, New London, CT, 1992.

C. T. Chen and F. J. Millero, "Speed of sound in seawater at high pressures," J. Acoustic. Soc. Am., 62(5), 1129-1135, 1977.

W. Hackmann, *Seek & Strike*, London: Her Majesty's Stationary Office, 1984.

E.E. Hofmann, L. J. Peitrafesa, and L. P. Atkinson, "A bottom water intrusion in Onslow Bay, North Carolina." Deep-Sea Research, 28 A, 329-345, 1981.

E. P. Jensen and F. Sabadini, "Low frequency active (LFA) bottom loss requirements," in SACLANTCEN Conference Proceedings, 24-28 May 1993, SACLANT Undersea Research Center, Serial no. CP-42, 1987.

B. W. Jones, "Measurement of time spread and energy spitting," in Proceedings of a Seminar on Active Sonar Signal Processing, 12 Dec 1989, Naval Undersea Systems Center, New London, Tracor Doc. T-90-01-9532-U, 1990.

R.S. Kennedy, *Fading Dispersive Communication Channels*, New York: John Wiley & Sons, 1969.

G.A. Kerr, P. J. Bucca, J. K. Fulford, and S. W. Snyder, "Environmental variability during the littoral warfare advanced development sponsored focused technology experiment (FTE 96-2)," NRL/MR/7182--97-8056, Naval Research Laboratory, Stennis Space Center, MS, 1997.

W.C. Knight, R. G. Pridham, and S.M. Kay, "Digital signal processing for sonar," Proceedings of the IEEE, 69 (11), 1451-1506, 1981.

S O'Keefe, F.B. Kelso, and C.E. Mundy "...From the sea," Navy and Marine Corps White Paper, 1992.

B. H. Pasewark, A. Al-Kurd, S. N. Wolf, T. C. Yang, "Acoustic transmission loss measurement," in Joint NAVSEA littoral warfare advanced development (LWAD)(SCV-97) quicklook briefing proceedings," Office of Naval Research, Arlington, VA, 1997,

G.A. Scanlon, "Estimation of bottom scattering strengths from measured and modeled AN/SQS-53C reverberation levels," Masters Thesis, Naval Postgraduate School, Monterey, CA, 1995.

R. J. Soukup and P. M. Ogden "Bottom backscattering measured off the South Carolina coast during littoral warfare advance development focused technology experiment 96-2," NRL/MR/7140--976-7905, Naval Research Laboratory, Washington D.C., 1997.

J. L. Stewart and M. K. Brandon, "Random medium? Correlation loss?," Paper presented at NATO-Marina Italian Advanced Study Institute on Stochastic Problems in Underwater Sound Propagation, Lerici, 1967.

A. Tanaka, "An analysis of energy spreading loss associated with tactical active sonar performance in a shallow water environment," Masters Thesis, Naval Postgraduate School, Monterey, CA, 1996.

J. Urick, *Principles of Underwater Sound*, 3rd ed., New York: McGraw Hill, 1983.

D. E. Weston, "Correlation loss in echo ranging," J. Acoust. Soc. Am., 37, 119-124, 1965.

A. F. Wittenborn, "Analysis of signal processing and related topics pertaining to the AN/SQS-26 sonar equipment," Summary Report III, Tracor Inc., Austin, Texas, 1965.

L. J. Ziomek, *Fundamentals of Acoustic Field Theory and Space-Time Signal Processing.* Ann Arbor: CRC Press, 1995.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center 8725 John J. Kingman R., STE 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, CA 93943-5101	2
3.	Chairman (Code OC/BF) Department of Oceanography Naval Postgraduate School Monterey, CA 93943-5100	2
4.	Commanding Officer Naval Research Laboratory Code 5100/5123 Washington, D.C. 23037-5000 Attn: Dr. T. C. Yang	1
5.	Officer In Charge Naval Research Laboratory Stennis Space Center, MS 39529-5000 Attn: Dr. S. Chin-Bing	1
6.	Dr. James H. Wilson Neptune Sciences, Inc. 3834 Vista Azul San Clemente, CA 92674	2
7.	Mr. Ed Jensen Naval Undersea Warfare Center Division 1176 Howell St. Newport, RI 02841-1708	1
8.	Program Executive Office (PMS -411) Surface Ship ASW Systems Washington, D. C. 20362 - 5104 Attn: CAPT R. Goldsby	1

9. Capt. G. Nifontoff 3
Naval Space and Warfare Command
PD 18
153560 Hull St.
San Diego, CA 92152-5002
Mr. Steve Payne PMW-185
CAPT Charles Hopkins PMW-185

10. Program Executive Office 1
Air ASW, Assault and Special Mission Programs (PMA - 299)
Suite 156, Unit # IPT
47123 Buse Rd.
Patuxent River, MD 20670-1547
Attn: LCDR Papa (Air - 4.5.1.2)

11. Lt. Peter E. Smith 1
Department Head Training
Surface Warfare Officers School Command
446 Cushing Road
Newport, RI 02841-1209